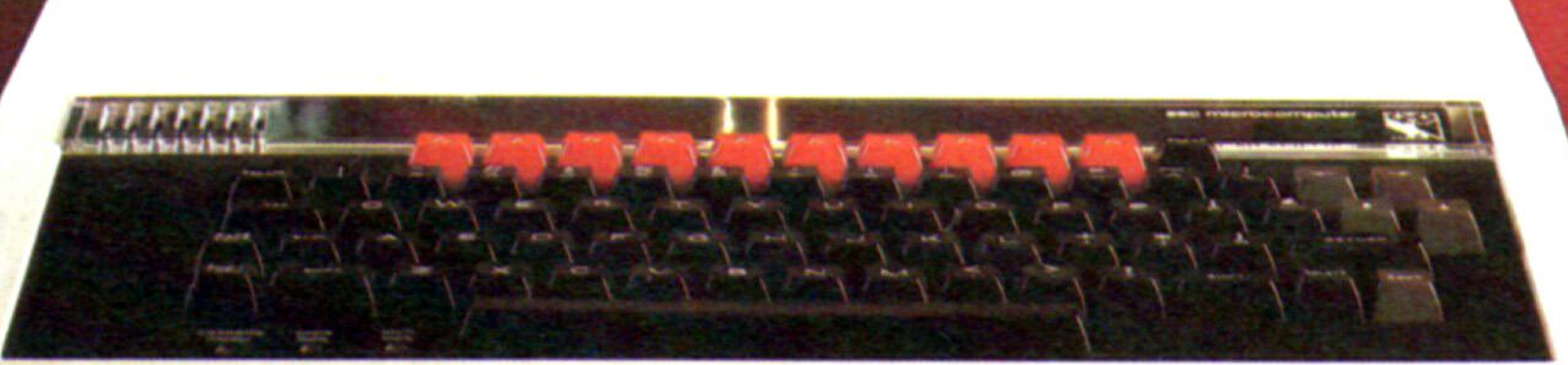


CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR







Editorial Delta, S.A.

150 ptas.

mi COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen IV - Fascículo 43

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-005-8 (tomo 4)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 078411
Impreso en España - Printed in Spain - Noviembre 1984

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.



La imagen ideal



Liz Heaney Software, cortesía de Pilot Software

La importancia de la presentación
Estos dos programas basados en disco cuestan alrededor de 9 000 pesetas cada uno, pero uno de ellos se ha empaquetado en una forma imaginativa, que atrae la atención y sugiere que vale la pena pagar su precio, mientras que el otro, en comparación, parece una caja demasiado convencional cuyo precio resulta muy elevado

Tanto en la comercialización de un nuevo ordenador como en la de un paquete de juegos es básico crear la "imagen del producto"

El *marketing* consiste en construir un puente entre el producto que los vendedores tienen que vender y el dinero del que disponen los potenciales compradores. Al diseñar un puente que estimule y resista la máxima cantidad de tráfico, los agentes de marketing tienen que hacer suposiciones (algunas veces basadas en prospecciones de mercado) sobre las necesidades, deseos y caprichos de su público, y después respaldar esas conclusiones con importantes inversiones.

Las decisiones relativas a la comercialización empiezan a tomar forma cuando la producción de una nueva máquina se halla en fase de planificación. Las funciones que tendrá la máquina, la cantidad de unidades que se fabricarán y cuánto se puede gastar en la producción de cada unidad, todos son factores que incidirán en la comercialización.

A la comercialización del software se le aplican consideraciones similares. No tiene sentido gastar grandes sumas de dinero desarrollando y vendiendo un juego que, para recuperar la inversión, resultará más caro de lo que se pueden permitir sus compradores potenciales. Pero una firma de software que escatime dinero en desarrollo se encontrará vendiendo un producto que se queda corto en prestaciones o que está plagado de errores, o ambas cosas a la vez, corriendo el riesgo, por lo tanto, de que su marca comercial se desprestigie, con un efecto potencialmente pernicioso para sus futuros productos. Por el contrario, si se invierte muchísimo dinero en desarrollo pero nada en comercialización, se encontrará vendiendo un producto espléndido pero que nadie conoce.

También es importante determinar ya en una etapa temprana el lugar que ocupa un producto en relación a artículos similares existentes en el mercado. Es aquí donde la "imagen" del producto adquiere una importancia vital. Los fabricantes de cigarrillos y de jabón en polvo comparten una dificultad: a la hora de la verdad, todos los productos se parecen mucho entre sí. Los fabricantes de hardware para ordenadores no se hallan exactamente en esta misma situación, pero no siempre es fácil explicar con precisión el carácter individual de una máquina a quienes acuden a adquirir una.

Por este motivo los fabricantes de hardware y de software, al igual que los fabricantes de cigarrillos y de jabón en polvo, recurren a la creación de una

"imagen del producto", una especie de proyección psicológica sintetizada del producto. Es más eficaz vender una idea que un mero producto, principio tan bien reflejado en el axioma publicitario que afirma que "vende más el olor que la salchicha".

El "diseño industrial" de la máquina (su aspecto exterior y el trazado exterior de sus interruptores y teclas de función) con frecuencia es un punto de partida útil para desarrollar la imagen de un producto de hardware. El BBC Micro, por ejemplo, es sobrio y utilitario, como corresponde a su publicidad, que subraya siempre su especial valor educativo. Las carcasas de otras máquinas suelen estar decoradas con molduras y logotipos llamativos para seducir al amante de los juegos. La gama Atari XL (cuyo nuevo estilo responde a un esfuerzo de marketing por rescatar al producto de la depresión de mediados de 1983) tiene un estilo austero, de bordes romos y aspecto militar; muy adecuado para jugar a *Tank battle* (Batalla de tanques). Commodore, en su campaña para introducirse en el mercado norteamericano, le encargó a Ferdinand Porsche, el famoso diseñador de automóviles, que mejorara la presentación de su gama con formas elegantemente redondeadas, que le dieran, con sutileza, un toque futurista.

Por el contrario, el aspecto físico del Sinclair Spectrum, con su pequeña carcasa negra y sus teclas multifunción cuidadosamente rotuladas, supone una máquina que ofrece mucho en muy poco espacio, con una clara sugerencia de que lo que proporciona supera con creces lo que vale.

La imagen de un producto va más allá de consideraciones sobre su aspecto físico. La imagen se debe difundir mediante una plataforma publicitaria coherente que refuerce la idea. La supuesta "memoria de elefante" del Commodore 64 se vende muy bien incluyendo un paquidermo en los anuncios en revistas y televisión. Para el BBC, el logotipo del búho evoca en la mente del público la idea de la "sabiduría".

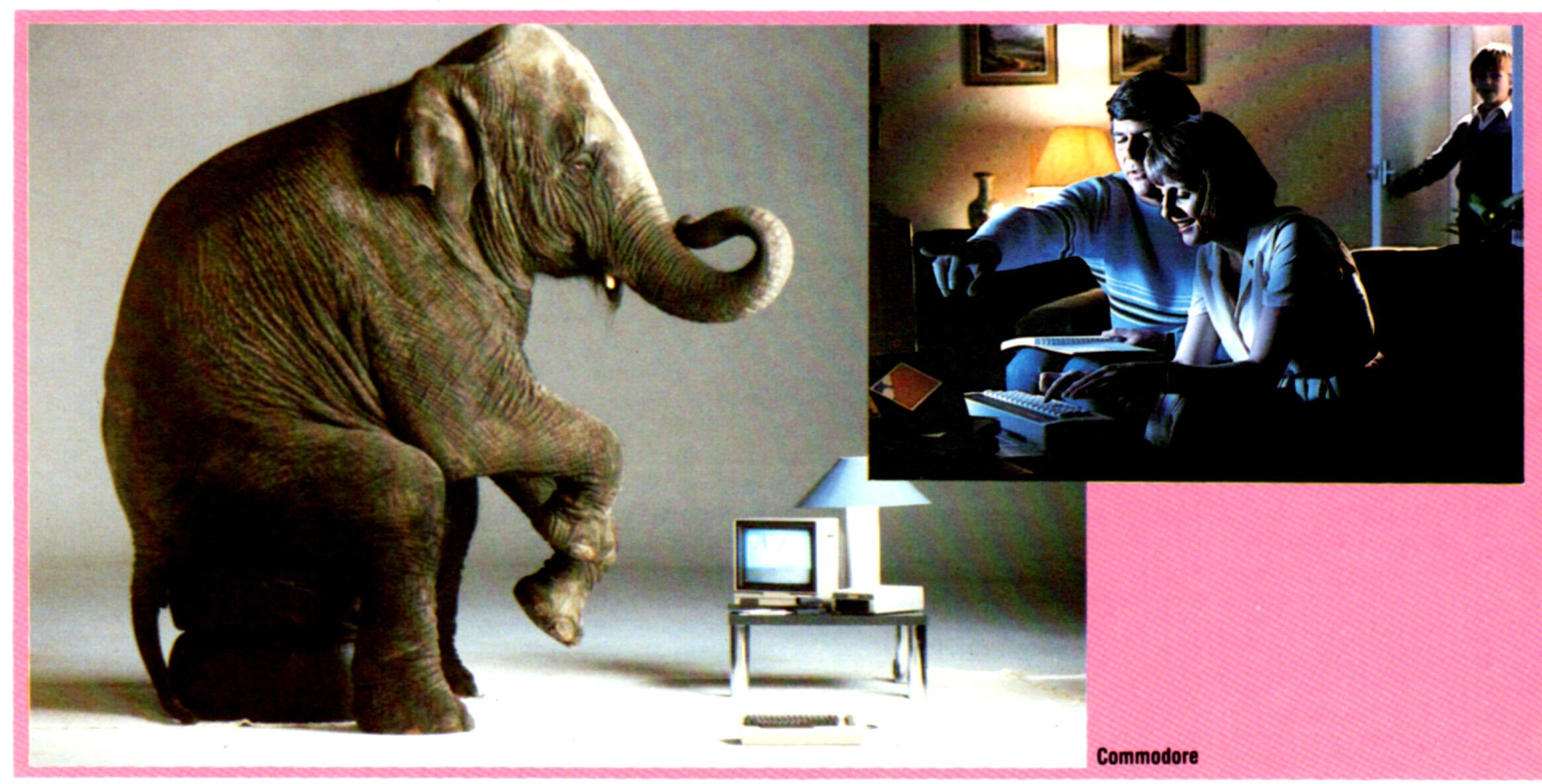
Naturalmente, el nombre es un punto importante. Los primeros ordenadores personales tenían que lidiar arduamente contra una imagen arraigada en la mente del público por las películas de los años sesenta y principios de los setenta en las que el ordenador aparecía invariablemente como un ente inhumano cuyo control escapaba a los simples mortales, a imagen y semejanza del "Hermano Mayor", el omnipresente dictador de *1984*, la novela de anticipación de George Orwell. Debido a ello a los micros se les asignaban nombres domésticos que deliberadamente rechazaban toda posible asociación con una tecnología elevada. De ahí el PET (mascota) y el Apple (manzana).

El Macintosh se anunció en Gran Bretaña y Estados Unidos mediante una insistente campaña de televisión filmada en aquélla por el director cinematográfico Ridley Scott, en la que se veía a una mujer haciendo añicos la pantalla de un "Hermano Mayor", para representar la recién hallada libertad que ofrecía el último producto de Apple. El slogan era "Gracias al Macintosh, 1984 no será como 1984". Para el mercado norteamericano, "Mackintosh" (con "k") es una variedad de manzanas.

Pero la gente quiere estar segura de que no se le está vendiendo un mero juguete. El PET encontró una gran resistencia en sus ventas en este punto cuando Commodore intentó desarrollar el mercado de gestión a finales de 1970. La primera línea de ataque fue sugerir que el nombre era un acrónimo de Personal Electronic Transactor (gestor electrónico personal), que sonaba mucho más científico, pero la táctica no resultó muy convincente y hubieron de recurrir a cambiar el nombre del micro por el de Commodore Business Computer.

La idea de la alta tecnología se puso de moda a medida que la gente se iba sintiendo cada vez más cómoda con la idea de disponer de ordenadores en su entorno. En esta categoría, la efectividad de los nombres de los productos depende de apartarlos lo más posible del lenguaje cotidiano, de modo que se

Imágenes
En marketing, la importancia de crear una imagen de producto poderosa se aprecia claramente en nuestro ejemplo: el paquidermo del Commodore sugiere la "memoria de elefante" del Commodore 64; las escenas hogareñas y escolares del BBC transmiten la flexibilidad, cordialidad e importancia educativa del Electron; mientras que el escenario de Apple, bastante más fantástico con sus alusiones orwellianas, les promete a los usuarios del Apple la liberación de los trabajos fatigosos en un mundo lleno de ordenadores humanizados. Sin embargo, al igual que la mayoría de las técnicas de gran poder, pueden tener efectos imprevistos: proverbialmente, al elefante lo aterrorizan los ratones (a diferencia del Macintosh y el Lisa) y podría sugerir un producto anticuado; a los potenciales compradores del Electron el ambiente familiar les podría resultar aburrido y estereotipado, y la asociación con la escuela, intimidante; los clientes de Apple podrían pensar que se los está retratando como clones sin inteligencia



Commodore



prefieren acrónimos o incluso grupos de letras que no quieren decir nada, antes que palabras que se encuentran en el diccionario. Las letras más raras, las que más puntos valen en el *scrabble* —juego que consiste en crear el mayor número de palabras cruzadas con el menor número de letras—, también son las que más valen aquí. Y de ahí máquinas como el ZX81, el MTX500 y el MZ-700.

El empaquetamiento, que en el caso del hardware es meramente protector, reviste una trascendental importancia en la imagen del software. En líneas generales, los vendedores de software pueden seguir dos estrategias: gastarse sólo un mínimo en empaquetamiento (la caja de la cassette con un folleto en color en su interior) y un correspondiente precio "de ganga", o construir el "valor percibido" del producto poniéndolo en una caja más grande, que a menudo tiene la apariencia de un libro, y añadirle extras.

La imagen se proyecta a través de la publicidad. En términos generales, la relación entre marketing y publicidad es la relación entre estrategia y táctica. Las cuestiones acerca de cuándo anunciar y a cuánto debe ascender el presupuesto de publicidad son decisiones de marketing.

Pero la promoción de la imagen del producto no siempre ayuda a vender cuando la imagen es mala. Hubo una campaña de comercialización para cigarrillos que ya es leyenda en el ambiente publicitario. El nombre comercial «Strand» se promocionó ampliamente en anuncios para cine y televisión, asociado a un hombre solitario enfundado en un impermeable blanco; el slogan rezaba: "Nunca estás solo si tienes un Strand". Pero el mensaje que se deducía era: "Los solitarios fuman Strand", y la marca no se vendió.

La imagen de algunos ordenadores existentes puede ser como un *boomerang* en este mismo sentido. El elefante del Commodore podría servir para recordar que la escritura de programas en el 64 es pesada.

Se podría decir que la creación y la proyección de las imágenes son el lado "creativo" de la comercialización (así lo creen en el ambiente publicitario). Pero la logística del marketing es igual de importante o quizá aún más, y con mucha frecuencia éste es el eslabón débil de la cadena. Una cosa es estimular la imaginación del público sobre un nuevo producto, y otra muy distinta, y más ardua, es introducirlo en sus hogares u oficinas.

¿Ha de realizarse la distribución en forma de pedidos por correspondencia, al estilo Sinclair? Ésta es una forma económica de hacerles llegar los productos a los clientes, pero ¿cómo les asegura un buen soporte? Probablemente habrá que vender a precios de ganga para contrarrestar la seguridad que proporciona comprar en una tienda, pero si se reducen demasiado los márgenes de beneficio podría haber dificultades para financiar la producción en masa, con las consiguientes demoras en la entrega.

Vender a través de las cadenas de tiendas ya establecidas les proporcionará a los clientes una fuerte sensación de seguridad en su compra, pero estos establecimientos comerciales exigirán la contrapartida de un mayor porcentaje para la tienda, reduciendo nuevamente los beneficios del fabricante. Un paquete de software cuyo precio de venta al público sea de 1 100 pesetas reportará menos de 500 pesetas después de que la cadena haya deducido su parte. También cabría pensar en instalar una red de distribuidores seleccionados, cada uno de ellos capaz de dar a sus clientes ayuda y consejo especiales sobre sus productos.

En la lucha por la supervivencia en el mundo de los micros, la desaparición de algunos fabricantes de excelente hardware y software ha puesto de manifiesto el hecho de que fabricar productos con frecuencia es menos de la mitad de la batalla. Donde la historia empieza realmente es en crear y mantener mercados. Y ahí es donde todo empieza a resultar difícil.



Acorn/BBC

Apple

Una flor de calidad

Las impresoras de rueda margarita realizan una labor eficaz y poseen características tan útiles como el espaciado proporcional

A primera vista, una impresora de rueda margarita podría parecer una compra extraña para el usuario de un ordenador personal. No es realmente adecuada para listar programas, es lenta y cuesta más que una impresora matricial. No obstante, para algunas aplicaciones es una buena elección. El área en la cual la impresora de rueda margarita gana mucho terreno es en la calidad de la impresión. Una impresora matricial construye cada carácter imprimiendo un patrón de puntos: independientemente de cuántas agujas se utilicen en el cabezal de impresión, los puntos individuales siempre se ven en el texto impreso.

Con una impresora de rueda margarita, por el contrario, los caracteres se producen mediante un conjunto de tipos que golpean contra una cinta entintada, al igual que en una máquina de escribir. Estos bloques de tipos, uno para cada carácter, están dispuestos en círculo, como los pétalos de una flor, de ahí su nombre: *rueda margarita*. La impresión es más fácil de leer y parece más profesional.

La contrapartida por esta superior calidad de impresión se encuentra en la velocidad de la máquina: las impresoras de rueda margarita son mucho más lentas que las matriciales de igual precio. El motivo de esta lentitud radica en sus distintos métodos de impresión. Para imprimir un carácter utilizando una impresora margarita, la rueda de impresión ha de girar primero hasta que el "pétalo" requerido quede en la parte superior; luego el martillo de impresión golpea el tipo seleccionado y el carro se desplaza para producir el carácter siguiente.

Compare esto con el funcionamiento de una impresora matricial, donde los puntos se imprimen a medida que el carro se desplaza a través del papel, y podrá comprender la diferencia de velocidad entre los dos tipos de impresora. Una rueda margarita suele imprimir alrededor de 20 caracteres por segundo (cps); una impresora matricial Epson FX-80 vale aproximadamente lo mismo, pero puede imprimir a una velocidad de 160 cps. Algunas impresoras de rueda margarita son más rápidas, pero su precio asciende a más del doble que el de las impresoras margarita normales.

Para aumentar la velocidad de impresión, tanto las impresoras de rueda margarita como las matriciales con frecuencia poseen dos características extras: impresión bidireccional y búsqueda lógica. *Bidireccional* significa simplemente que una línea de texto se imprime de izquierda a derecha y la línea siguiente se imprime de derecha a izquierda. La impresora no ha de esperar que el carro retorne y, por lo tanto, imprime más rápido. *Búsqueda lógica* significa que el carro se salta espacios para llegar a la siguiente palabra del texto; las impresoras menos sofisticadas tardan el mismo tiempo en "imprimir" un espacio que cualquier otro carácter.

Las impresoras matriciales tienen las formas de sus caracteres almacenadas en memoria ROM dentro de la impresora; las impresoras de rueda margarita tienen el carácter almacenado como bloques de tipos en la rueda de impresión. Cada método posee sus ventajas: con una impresora matricial las formas de los caracteres se pueden redefinir enviándole a la impresora, desde el micro, códigos de escape adecuados. Con una impresora de rueda margarita el proceso es mucho más sencillo: simplemente se cambia la rueda corriente por una distinta.

Las ruedas margarita vienen en diversos estilos y espaciados de tipos; el *estilo de tipos* alude al diseño de los caracteres, y el *espaciado* a su anchura. Algunos de los estilos de tipos más comunes son el Courier, Romano, Gótico y cursiva. El espaciado normalmente es de 10 o 12 caracteres por pulgada. Una rueda margarita puede ser de plástico o de metal. Las metálicas poseen la ventaja, respecto a las plásticas, de que tienen una vida útil mucho más larga, pero son mucho más caras que éstas.

Sin embargo, estos diferentes estilos presentan el inconveniente de que pocos de ellos son exactamente iguales que el juego de caracteres que utiliza un micro. Esto significa que algunos de los caracteres del teclado de su micro se imprimirán como algo totalmente diferente. A menudo el carácter "numérico" (#) imprime un signo de libras (£), o un corchete ([) se imprime como una fracción (½). En muchos estilos de tipos el número "0" (cero) no se diferencia de la letra "O" mayúscula y, del mismo modo, la "l" minúscula se puede tomar por un número "1". Mientras que las impresoras de rueda margarita más caras poseen ruedas de impresión de 127 caracteres, la mayoría sólo pueden imprimir 92 o 96 caracteres y son éstas las que sufren más este tipo de problema.

Como usted se puede imaginar, tratar de depurar un programa no resulta nada sencillo si uno no está seguro de qué caracteres son "unos" y cuáles son "eles". Por esta causa, y también debido a su lentitud, una impresora de rueda margarita no es aconsejable si el usuario utiliza su ordenador básicamente para programar.

Efectos especiales

Una impresora de rueda margarita se puede programar para que produzca diversos efectos especiales, al igual que las impresoras matriciales. A pesar de que el número de estos efectos es limitado, el método para programar la impresora es idéntico al que se utiliza para una impresora matricial, es decir, enviando códigos de escape (véase p. 804). Por ejemplo, en una impresora de rueda margarita Diablo, el código ESC-E activa el subrayado automático y ESC-R lo desactiva. Utilizando el BASIC Mi-



crosoft estándar, se entraría LPRINT CHR$(27);"E"; y LPRINT CHR$(27);"R"; para enviar a la impresora los códigos antes mencionados. Otros códigos incluyen ESC-1 para fijar una tabulación; ESC-9 para fijar el margen izquierdo; y ESC-U para hacer saltar el papel media línea hacia arriba (útil para subíndices). En la rueda margarita, el equivalente de "enfatizar" en una impresora matricial se denomina *destacar* (cada carácter se imprime cuatro veces para que resalte).

Algunas impresoras de rueda margarita permiten que se varíe tanto la distancia entre caracteres como el espaciado de líneas. En estos casos se puede utilizar la impresora para crear imágenes gráficas o vuelcos de pantalla, al igual que en una impresora matricial (véase p. 824). Si un pixel de la pantalla está "encendido", la rueda margarita imprime un punto; si está "apagado", entonces se imprime un espacio. Reduciendo la distancia entre caracteres se puede imprimir sobre papel una línea horizontal completa de la pantalla. De forma similar, el espaciado entre líneas se puede reducir hasta no dejar ningún espacio vacío entre una y otra línea. El proceso, sin embargo, es muy lento.

Una característica de la que carecen las impresoras matriciales es la capacidad de centrar títulos automáticamente. Enviándole ESC= a una impresora Diablo, ésta centrará el resto de esa línea entre los márgenes. Otra característica novedosa es el tabulador decimal: ESC−H hará que todos los números se impriman con las comas decimales alineadas, lo cual es muy útil para sumas de dinero.

Las impresoras de rueda margarita más caras por lo general pueden efectuar espaciado proporcional. Con esta característica el espaciado no es el estándar de 10 o 12 caracteres por pulgada, sino que varía a tenor de la anchura del carácter que se esté imprimiendo. Por ejemplo, el carácter "w" es mucho más ancho que el carácter "i", de modo que con el espaciado proporcional el carro no se iría tan lejos para una "i" como lo haría para una "w". Esto significa que los caracteres de las sucesivas líneas de texto no quedan exactamente uno debajo del otro, y el efecto global resulta visualmente más agradable. Las líneas de texto que usted lee en este momento están espaciadas proporcionalmente.

En el caso de una oficina, la necesidad de una impresora de rueda margarita puede estar plenamente justificada; para la mayoría de los usuarios de ordenadores personales, probablemente no. Existe, sin embargo, una alternativa: adaptar una máquina de escribir electrónica. Las impresoras de rueda margarita cuestan más que las máquinas de escribir electrónicas, a pesar de que ambas utilizan el mismo procedimiento de impresión. No obstante, hasta hace muy poco tiempo las máquinas de escribir no se podían conectar fácilmente con un ordenador: carecían de circuitos de interface o de conector RS232. Pero en la actualidad todas las máquinas de escribir electrónicas que existen en el mercado vienen con una interface para ordenador incorporada o bien se pueden equipar con un kit de interface. La gran ventaja de adaptar una máquina de escribir electrónica, aparte del ahorro de dinero que representa respecto a una impresora de rueda margarita comparable, es que se la puede seguir utilizando como una máquina de escribir convencional. Así, usted posee tanto una máquina de escribir como una impresora de rueda margarita.

La calidad tiene su precio

Imprimir muestra con 10 caracteres por pulgada
Imprimir muestra con 12 caracteres por pulgada
Imprimir muestra con 15 caracteres por pulgada

Para subíndices se utiliza el salto de media línea:

H_2O

ESC-E activa el subrayado automático y ESC-R lo desactiva.

La impresión destacada hace que el texto **resalte** del resto del texto.

El código ESC= se utiliza para centrar texto:

Un título centrado

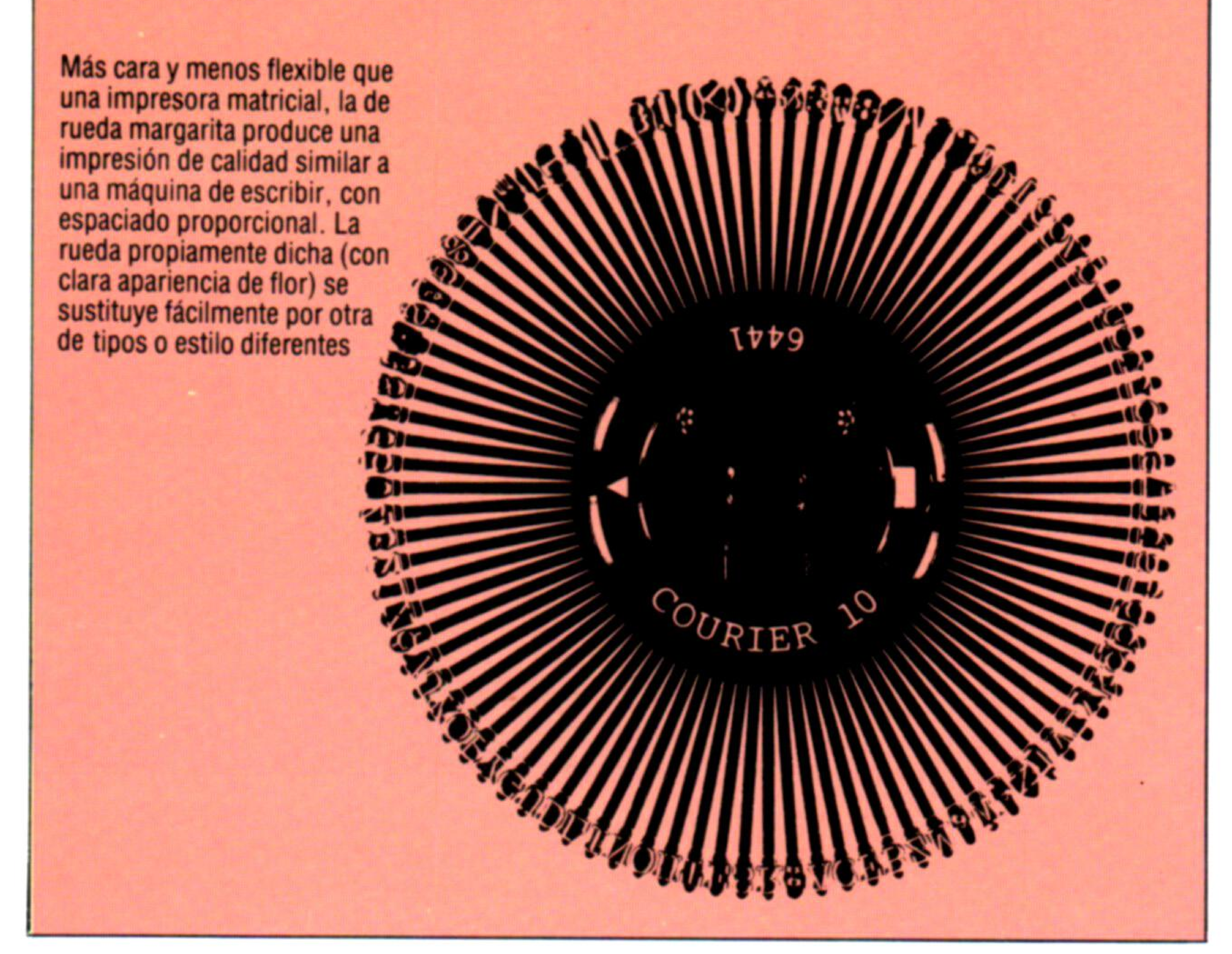
Estas líneas de texto se imprimieron sin la facilidad de espaciado proporcional. Observe especialmente cómo quedan espaciados los números 0 1 2 3 4 5 6 7 8 9. Sin el espaciado proporcional, la anchura de cada carácter es la misma. Por ejemplo, una "w" tiene el mismo ancho que una "i":

wwwwwwwwwwwww
iiiiiiiiiii

Estas líneas de texto se imprimieron utilizando la facilidad de espaciado proporcional. Observe especialmente cómo quedan espaciados los números 0123456789. Utilizando el espaciado proporcional, la anchura de cada carácter varía. Por ejemplo, una "W" es mucho más ancha que una "i":

wwwwwwwwwwwww
iiiiiiiiiii

El efecto global es que el texto resulta más atractivo.



Más cara y menos flexible que una impresora matricial, la de rueda margarita produce una impresión de calidad similar a una máquina de escribir, con espaciado proporcional. La rueda propiamente dicha (con clara apariencia de flor) se sustituye fácilmente por otra de tipos o estilo diferentes

Leer en el teclado

Aquí le proporcionamos un programa sencillo para mover un "lápiz de gráficos" desde el teclado del Spectrum

El BASIC le permite al usuario entrar la información desde el teclado utilizando INPUT e INKEY$. La sentencia INPUT lee un número o serie de caracteres, terminados por ENTER, sobre una variable numérica o de tipo string (o alfanumérica). La función INKEY$ explora el teclado y devuelve o bien un string conteniendo el carácter de la tecla pulsada o, si no se ha pulsado ninguna, un string vacío. Se trata de dos instrucciones muy útiles, pero para algunas aplicaciones especiales (por ejemplo, cuando se deben leer simultáneamente combinaciones de teclas) la función IN se puede utilizar para leer directamente el teclado.

Las teclas del Spectrum están conectadas al microprocesador Z80 a través de puertas de entrada-salida. Hay 65 536 puertas de entrada-salida y cada una de ellas se puede direccionar individualmente. De la misma manera en que se usan PEEK y POKE para leer o escribir en la memoria, IN y OUT se utilizan para leer y escribir en las puertas de entrada-salida. La función IN(m) devolverá el valor de la puerta m, mientras que la sentencia OUT(m,n) escribirá el valor n en la puerta m.

El teclado se compone de cuatro filas de 10 teclas, estando dividida cada fila en medias filas de cinco teclas. Cada media fila corresponde a una puerta, como refleja la ilustración. Observe que las teclas de las medias filas izquierdas se corresponden con sus puertas de derecha a izquierda, mientras que las medias filas de la derecha se corresponden de izquierda a derecha.

Los cinco bits más hacia la derecha de una puerta se corresponden cada uno de ellos con una tecla, y tomarán el valor 0 si se ha pulsado la tecla que les corresponde, o 1 en caso contrario. Los bits marcados con una X son irrelevantes: pueden contener cualquier valor binario. Esto implicaría que el valor de IN(m) quedaría indeterminado. Este problema se puede superar poniendo por software el valor de los tres bits superiores a cero, mediante la utilización de la siguiente instrucción:

```
DEF FN a(p) = p - INT(p/32)*32
```

donde p es el valor de la puerta (el byte indeterminado devuelto por IN(m). La división p/32 dará un número desde 0 hasta justo por debajo de 8. Tomando INT(p/32) se descartará la fracción decimal de este número, dejando un valor entero entre 0 y 7. Multiplicando éste por 32 se obtiene luego un valor que representa los tres bits superiores (más hacia la izquierda) de p. Restándoselos al propio p

En contacto
El teclado del Spectrum está formado por ocho bloques de cinco teclas, y cada bloque se controla constantemente mediante un byte exclusivo o "puerta". Cada tecla se corresponde con un bit de su puerta; mientras se mantiene pulsada una tecla, su bit "de señal" pasa de uno (su estado normal) a cero. Aquí se están pulsando las teclas Y, I y O, de modo que el valor binario de su puerta, el byte 57342, es XXX01001 (los bits del 5 al 7 son irrelevantes). De forma similar, se están pulsando A y S, de modo que el valor del byte 65022 es XXX11100



se eliminan estos tres bits, dejando un valor entero entre 0 y 31, que representa los cinco bits inferiores (más hacia la derecha) de p. El valor de FN a(p), por consiguiente, siempre se definirá con los tres bits superiores a cero, cualquiera que sea el valor de los tres bits superiores de p.

Por ejemplo, si se pulsara la tecla V, la puerta 65278 contendría:

X	X	X	0	1	1	1	1

FNa(IN(65278)), por lo tanto, retornará el valor:

0 0 0 0 1 1 1 1 = 15

Si se pulsara al mismo tiempo Caps Shift y la letra V, la puerta contendría:

X	X	X	0	1	1	1	0

y FN a(IN(65278)), por consiguiente, devolvería el valor 14. Si no se pulsara ninguna tecla, la puerta devolvería el valor 31 (00011111).

Pruebe este programa:

```
10 REM Imprimir el valor enmascarado de las
   puertas
20 REM Definir la función de máscara
30 DEF FN a(p) = p−INT(p/32)*32
40 REM Imprimir las puertas
50 PRINT AT 1,1;"Puerta              valor
   enmascarado"
60 PRINT "32766",FN a(IN(32766))
70 PRINT "49150",FN a(IN(49150))
80 PRINT "57342",FN a(IN(57342))
90 PRINT "61438",FN a(IN(61438))
100 PRINT "63486",FN a(IN(63486))
110 PRINT "64510",FN a(IN(64510))
120 PRINT "65022",FN a(IN(65022))
130 PRINT "65278",FN a(IN(65278))
140 FOR i = 1 TO 250:NEXT i
150 CLS
160 GO TO 50
```

Después de pulsar RUN, intente oprimir algunas teclas y observe cómo cambian los valores enmascarados de las puertas. Intente predecir los valores que se visualizarán para las distintas teclas y combinaciones de teclas. Si pulsara las teclas del cursor (5, 6, 7 y 8) obtendría los siguientes valores enmascarados en las puertas 61438 y 63486 respectivamente (puerta A y puerta B):

	Puerta A (puerta 61438)	Puerta B (puerta 63486)
←	31	15
↓	15	15
↑	23	15
→	27	15

Si pulsara la tecla Caps Shift, obtendría el valor enmascarado 30 para la puerta 65278 (a la que nos referiremos como puerta C).

Consideremos el sencillo programa para gráficos que ofrecemos aquí. Este programa leerá las teclas del cursor para trazar líneas horizontales, verticales y oblicuas, y, en caso de pulsarse la tecla Caps Shift, para desplazar un "lápiz para gráficos" sin dibujar ninguna línea. Las líneas oblicuas se trazan pulsando dos teclas del cursor al mismo tiempo.

En el programa, la línea 30 define una función (FN(a)) para poner a cero los tres bits superiores, como antes. La posición horizontal del lápiz para gráficos se representa mediante *x*, siendo *y* su coordenada vertical. La línea 50 establece la posición inicial en la que se encontraba el lápiz en el centro de la pantalla.

Las líneas 70 y 80 leen las puertas que se corresponden con las teclas del cursor. La media fila de 6 a 0 incluye tres de las teclas del cursor y se corresponden con la puerta 61438 (puerta A). La media fila de 5 a 1 contiene la tecla del cursor con la flecha hacia la derecha y se corresponde con la puerta 63486 (puerta B). La línea 90 lee la tecla Caps Shift: la media fila desde V a Caps Shift corresponde a la puerta 65278 (puerta C).

Entre la línea 110 y la 180 se verifican las ocho combinaciones "legales" de las teclas del cursor y se ajusta la posición *x,y* del lápiz para gráficos:

La línea 110 comprueba si ↑
La línea 120 comprueba si ↑ y →
La línea 130 comprueba si →
La línea 140 comprueba si ↓ y →
La línea 150 comprueba si ↓
La línea 160 comprueba si ↓ y ←
La línea 170 comprueba si ←
La línea 180 comprueba si ↑ y ←

Entre las líneas 200 y 240 se asegura que el lápiz no se desplace fuera de los límites de la pantalla. Por último, de no haberse pulsado Caps Shift, la línea 250 trazará en la pantalla la nueva posición del lápiz para gráficos. La línea 260 cierra el bucle para repetir nuevamente todo el proceso.

Un asunto cambiante
El programa emplea las técnicas analizadas para detectar múltiples pulsaciones de teclas: las teclas de flecha sin Caps Shift simultáneo permiten trazar líneas horizontales, verticales y oblicuas (oblicua=horizontal+vertical), mientras que las teclas pulsadas junto con Caps Shift producen los mismos movimientos del cursor sin trazado en pantalla. En un próximo capítulo utilizaremos las mismas técnicas para conseguir mejores efectos

Detección de pulsaciones múltiples de teclas

```
20 REM Preparar funcion para enmascarar los tres bits superiores
30 DF FN a(p) = p − INT (p/32)*32
40 REM Inicializar posicion lapiz
50 LET x = 127: LET y = 35
60 REM Leer puertas y eliminar tres bits superiores
70 LET puerta A = FN a(IN 61438)
80 LET puerta B = FN a(IN 63436)
90 LET puerta C = FN a(IN 65278)
100 REM Alterar posicion lapiz segun la tecla pulsada
110 IF puerta A = 23 AND puerta B = 31 THEN LET y = y+1
120 IF puerta A = 19 AND puerta B = 31 THEN LET x = x+1:
    LET y = y+1
130 IF puerta A = 27 AND puerta B = 31 THEN LET x = x+1
140 IF puerta A = 11 AND puerta B = 31 THEN LET x = x+1:
    LET y = y−1
150 IF puerta A = 15 AND puerta B = 31 THEN LET y = y−1
160 IF puerta A = 15 AND puerta B = 15 THEN LET x = x−1:
    LET y = y−1
170 IF puerta A = 31 AND puerta B = 15 THEN LET x = x−1
180 IF puerta A = 23 AND puerta B = 15 THEN LET y = y+1:
    LET x = x−1
190 REM Evitar que el lapiz se salga de la pantalla
200 IF x < 0 THEN LET x = 0
210 IF x > 255 THEN LET x = 255
220 IF y < 0 THEN LET y = 0
230 IF y > 175 THEN LET y = 175
240 REM Trazar punto si no se ha pulsado CAPS SHIFT
250 IF puerta C = 31 THEN PLOT x,y
260 GO TO 70
```

Test en cascada (2)

Continuando con este interesante test que iniciamos en el capítulo anterior, en esta ocasión introduciremos un nuevo elemento

En el capítulo anterior, refiriéndonos al ejemplo de los días de la semana, se hacía alusión a cómo se eliminaba una comparación; en aquel caso, se hacía por lógica, ya que sabemos que los días de la semana son siete. Por deducción, podría realizarse la misma operación, por ejemplo, con los meses del año, o bien con los dedos de la mano. Pero ello ocurre porque pertenecen a unas series limitadas cuyo número de elementos ya está establecido y es fijo; por ello, no podrían considerarse de igual modo otros casos cualesquiera.

Así, en estos casos nos encontramos con que deben controlarse unos valores determinados, pero variables, para poder realizar la eliminación de una pregunta tras la corriente sucesiva de otras. Por este motivo se deberá incluir una pregunta general al principio de la serie para que, de este modo, el dato entrado sea convenientemente filtrado antes de ser comparado.

El siguiente ejemplo lo desarrollaremos de dos formas: una es la que vemos representada a la izquierda; en ella no se ha tenido en cuenta ese filtro inicial, realizándose el mismo por seguimiento de la secuencia. En el próximo capítulo del apartado de Diagramación realizaremos la representación con dicho filtro.

Representación sin filtro inicial

Entradas dos cantidades, debe entrarse una tercera, que hace las veces de código y que, por otra parte, también marca la operación que debe realizarse con las dos primeras:

1: suma
2: resta
3: multiplicación
4: división

Conviene recordar que hay que efectuar un control para que la segunda cantidad, caso de tratarse de un código 4 (división), no sea cero, ya que ello nos daría un error. En este ejemplo, el test de eliminación, el filtro que determina si el número es válido, se lleva a cabo solamente tras haber pasado por todas las preguntas y haberse comprobado que no se puede optar por ninguna salida. Esto se considera válido, aunque no del todo recomendable, ya que supone una pérdida de tiempo y de trabajo, al tenerse que cuestionar cada control. En el caso de que se tratara de una cantidad mayor de decisiones, sería necesario comparar con todas ellas antes de llegar a la conclusión de que el código que se había entrado era erróneo.

Una gran ampliación

El Torch Disk Pack permite ampliar el BBC Micro, convirtiéndolo en un auténtico ordenador de oficina

El BBC Micro se puede ampliar para responder a unas especificaciones de gestión completas. Si bien esta capacidad se incorporó en la máquina desde el principio, la primera empresa que hizo uso de la idea no fue el fabricante, Acorn, sino Torch. El Torch Disk Pack permite que el usuario del BBC utilice la amplia gama de software que emplea el sistema operativo CP/M. Así y todo, el paquete completo vale apenas un poco más que un par de unidades de disco Acorn.

El sistema operativo VP/M requiere un microprocesador Z80 y unidades de disco. El BBC utiliza como microprocesador un 6502, pero el Torch Disk Pack le agrega un Z80, dejando que el 6502 del BBC maneje todas las funciones de entrada y salida. El 6502 controla el teclado, la pantalla y las unidades de disco, mientras el Z80, con sus 64 Kbytes de RAM, ejecuta los programas CP/M y, por regla general, "se hace cargo" del sistema. Los datos se pasan sucesivamente, una y otra vez, de un microprocesador a otro a través de la puerta de ampliación o "tubo" del BBC.

El procesador Z80 y sus 64 Kbytes de RAM se suministran en una pequeña placa de circuito impreso que se instala en el interior del BBC. El sistema combinado no utiliza la fuente de alimentación eléctrica de éste. En cambio, se conecta al BBC una nueva fuente de alimentación, situada en la caja principal del Disk Pack. La caja principal de éste contiene, asimismo, las dos unidades de disco flexible. Éstas son unidades de doble cara de 80 pistas, con unas características muy parecidas a las unidades de 800 Kbytes existentes para el BBC que fabrica Acorn.

Torch también ofrece un sistema alternativo. El ZEP 100 es esencialmente un Torch Disk Pack sin las unidades de disco y está pensado para aquellos usuarios de un BBC que ya posean sus propios discos. Torch también comercializa una gama de ordenadores de oficina que se componen de una placa con el BBC Micro más las unidades de disco, con una pantalla y un modem incorporados.

El Disk Pack está diseñado para instalarse debajo del BBC, conectándose ambos mediante cables cortos. La caja posee exactamente las mismas dimensiones que el ordenador, de modo que el sistema completo tiene un aspecto impecable. Lamentablemente, el Disk Pack hace que el teclado quede levantado unos 7 cm encima del escritorio, lo que puede dificultar un poco la digitación en el sistema completo.

El Torch Disk Pack viene equipado con todos los elementos necesarios para mejorar el BBC, con una importante excepción: no incluye el juego de chips para interface de disco que precisa el BBC para utilizar una unidad de disco. Algunas personas ya las habrán pagado al adquirir un BBC Micro, pero la mayoría se habrán comprado un BBC Micro Modelo B estándar y, por tanto, tendrán que desembolsar una cantidad adicional para añadir los chips de interface. Los BBC Micro modelo A se deben ampliar hasta el modelo B para que puedan emplear el Torch Disk Pack.

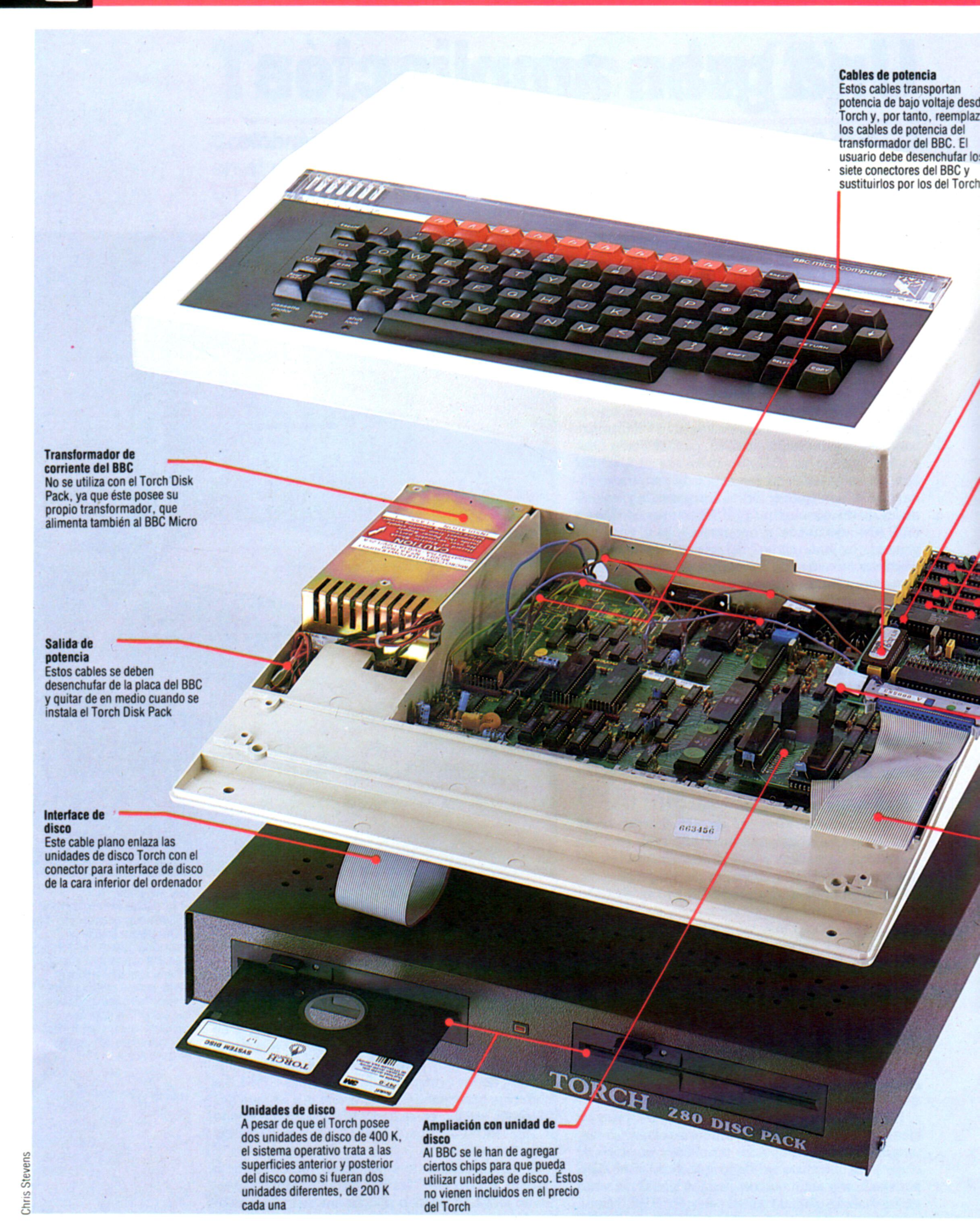


Chris Stevens

BBC mejorado
El Torch Disk Pack convierte al BBC Micro en un ordenador de oficina porque incluye un procesador Z80 con 64 K de memoria que permite utilizar un sistema operativo similar al CP/M. Incluye asimismo dos unidades de disco de 400 K y con todo ello el sistema completo vale apenas un poco más que un par de unidades de disco Acorn

Una vez que el Torch Disk Pack está correctamente instalado y el sistema encendido, funciona directamente (en modalidad de visualización 3) como un ordenador CP/M. El sistema operativo en disco utilizado está escrito por Torch y se denomina MCP. Éste es muy parecido al CP/M y ejecutará la mayoría de los centenares de programas CP/M. La diferencia principal entre los sistemas a los que nos referimos estriba en que el MCP está almacenado en una ROM dentro del BBC y no es necesario cargarlo en el micro.

El MCP dispone de muchas instrucciones que a los usuarios del BBC les resultarán familiares. Mediante MODE se pueden seleccionar distintas modalidades de visualización, y también están todas las



Cables de potencia
Estos cables transportan potencia de bajo voltaje desd Torch y, por tanto, reemplaz los cables de potencia del transformador del BBC. El usuario debe desenchufar los siete conectores del BBC y sustituirlos por los del Torch

Transformador de corriente del BBC
No se utiliza con el Torch Disk Pack, ya que éste posee su propio transformador, que alimenta también al BBC Micro

Salida de potencia
Estos cables se deben desenchufar de la placa del BBC y quitar de en medio cuando se instala el Torch Disk Pack

Interface de disco
Este cable plano enlaza las unidades de disco Torch con el conector para interface de disco de la cara inferior del ordenador

Unidades de disco
A pesar de que el Torch posee dos unidades de disco de 400 K, el sistema operativo trata a las superficies anterior y posterior del disco como si fueran dos unidades diferentes, de 200 K cada una

Ampliación con unidad de disco
Al BBC se le han de agregar ciertos chips para que pueda utilizar unidades de disco. Éstos no vienen incluidos en el precio del Torch

Chris Stevens

ROM/CCCP
El CCCP (Cambridge Console Command Processor) forma parte del MCP, el sistema operativo de Torch. El resto está en un chip en uno de los conectores de ROM del BBC

Microprocesador Z80
Éste asume el puesto del microprocesador 6502 del BBC Micro. Permite utilizar el sistema operativo Torch MCP. Dado que éste es similar al CP/M, se puede emplear una amplia gama de programas de gestión

RAM
Esta memoria es exclusiva para el procesador Z80, dejando los 32 K del BBC para utilizar como memoria de pantalla

Lengüetas de fijación
Estas lengüetas adhesivas pegan la placa del procesador Z80 en el interior de la carcasa del BBC Micro

Cable de conexión del Z80
Este cable plano conecta la placa del procesador Z80 con el BBC Micro a través de la interface "tubo"

instrucciones VDU y *FX. *KEY se utiliza para definir las teclas de función. Cuatro de las teclas de función están ya definidas con una selección de instrucciones de uso habitual.

Hay muchas otras instrucciones asociadas con la carga y tratamiento de archivos en disco. Aparte de ser capaz de cargar programas en código máquina sencillamente digitando sus nombres, el sistema ofrece una instrucción para cargar archivos específicos que construyen imágenes en el disco empleando el sistema operativo del BBC. PRINT saca un archivo de textos por una impresora, mientras que TYPE lo hace por pantalla. COMMAND utiliza un archivo como una serie de órdenes para el ordenador de modo similar a lo que hace EXEC en el BASIC BBC.

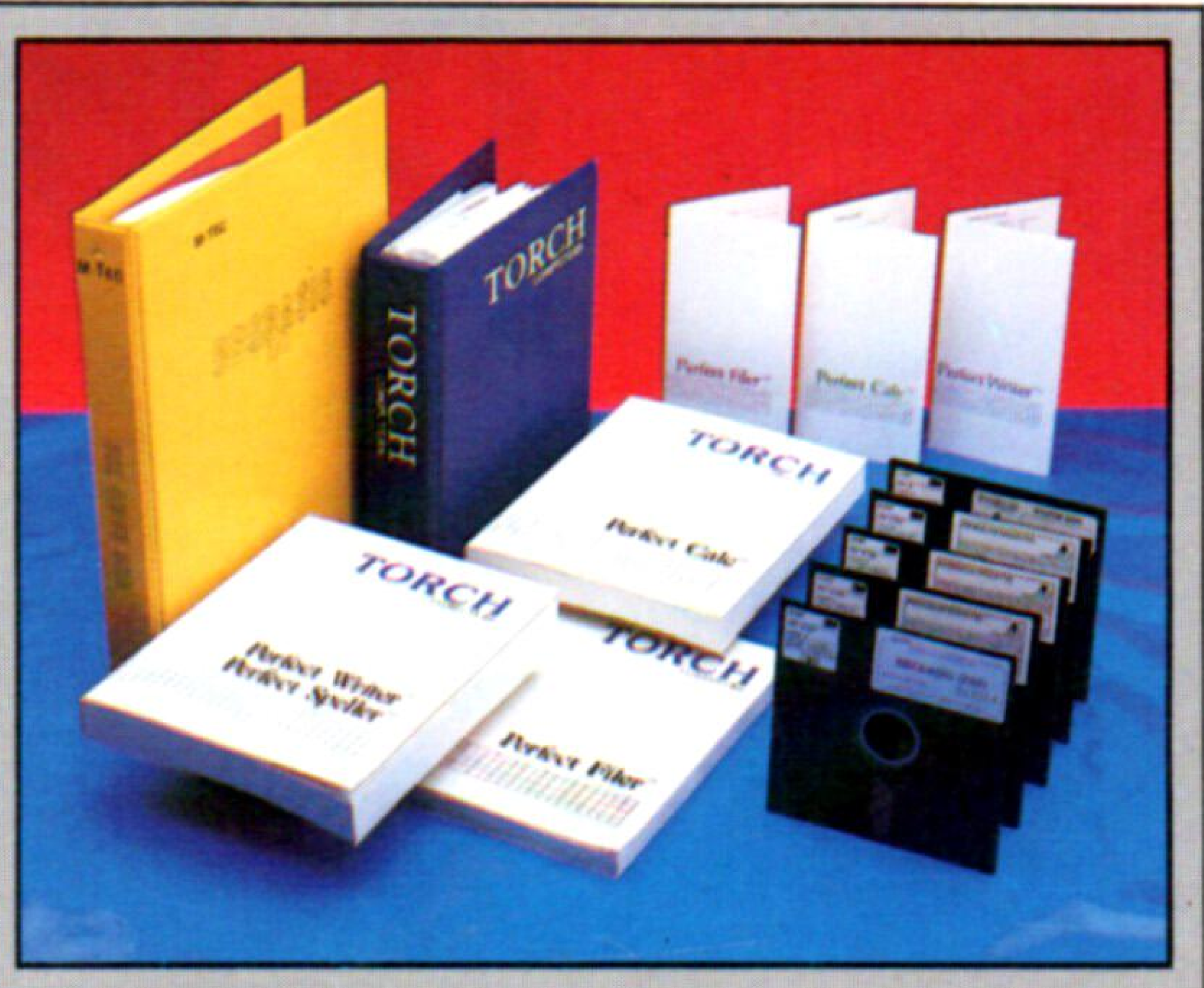
Existen otros varios programas de utilidad que residen en un disco de sistema que acompaña al Disk Pack y se cargan en la máquina tecleando el nombre correspondiente. Éstos incluyen una rutina para cambiar el diseño de los caracteres en la pantalla, una utilidad para escribir música, un *debugger* de código máquina, un editor de disco y una utilidad para permitir que el Torch lea discos Acorn y viceversa. Esta última es muy práctica, ya que los formatos que emplean ambos sistemas son diferentes. Esta utilidad permite, por ejemplo, crear un archivo de texto ejecutando un programa en BASIC BBC y luego editarlo con un programa CP/M para tratamiento de textos.

Si bien el formato del disco Torch es diferente del formato del Acorn, emplea la misma curiosa convención de tratar a las dos unidades de disco como si fueran cuatro unidades distintas.

A pesar de todas estas mejoras, el sistema todavía se puede utilizar como un BBC Micro estándar. Tecleando *BASIC el sistema ignora todos los extras añadidos por el Disk Pack. Extrañamente, el BBC Micro que usted debe tener está preparado para emplear cassettes y no discos. A pesar de que ahora está totalmente equipado para utilizar las unidades de disco del Disk Pack, sin necesidad de recurrir al MCP, el usuario debe especificar que se requiere el sistema de archivo en disco. Desde el BASIC BBC, se puede pasar a emplear de nuevo el Z80 en cualquier momento apagando y encendiendo la máquina, o bien tecleando *MCP.

El BBC Micro es un ordenador personal que dispone de muchas facilidades, pero *sólo* es un ordenador personal. Para utilizar un micro para gestión se requiere muchísimo más que lo que puede ofrecer el BBC solo. Cuando se le añade a un BBC Micro un Torch Disk Pack, se sigue teniendo allí el ordenador personal, listo para emplearlo al momento; pero, gracias a haberse producido esta agregación, el ordenador posee entonces todos los extras necesarios para convertirse en una auténtica máquina de oficina.

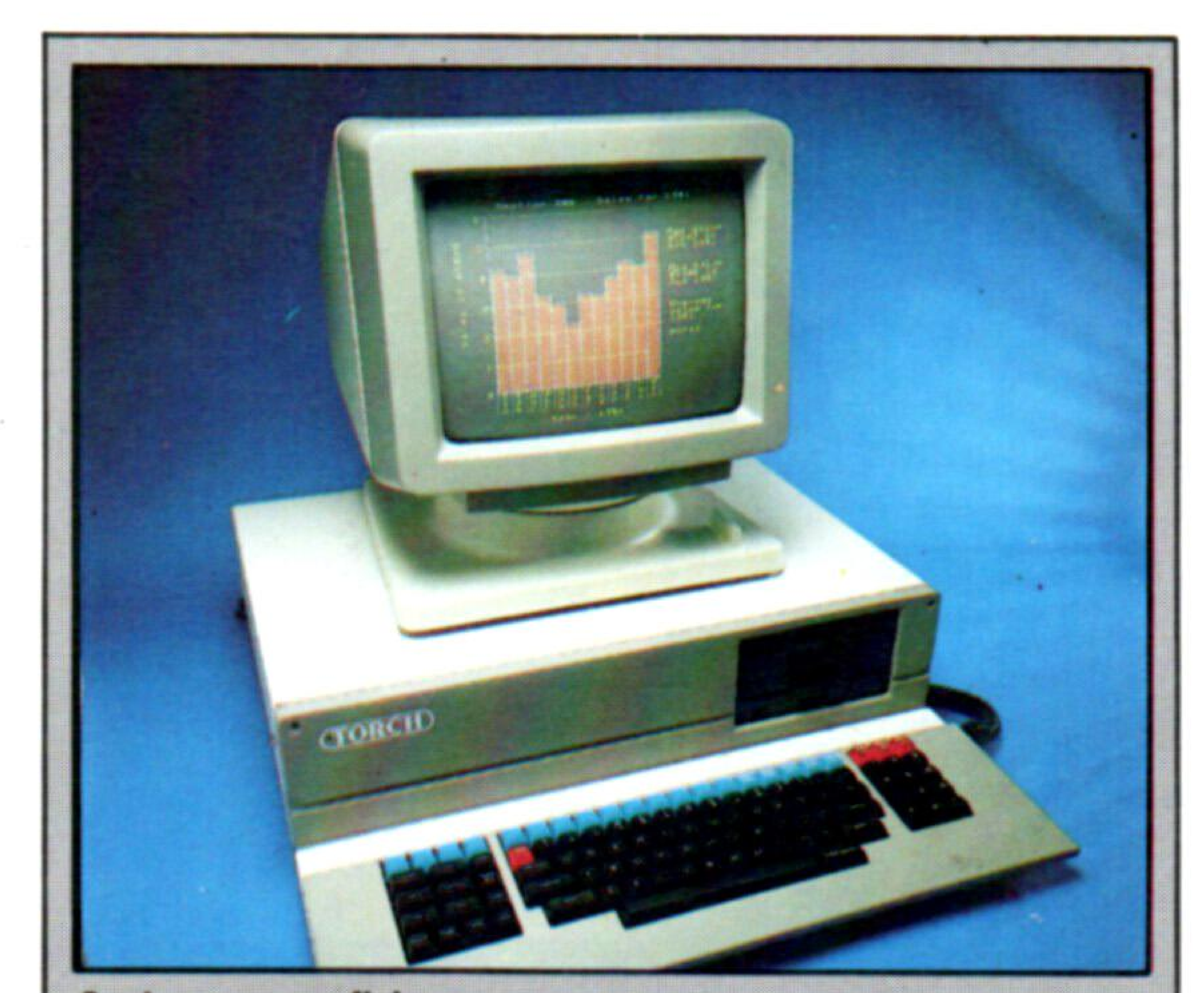
Instalando el Disk Pack
Instalar un Torch Disk Pack en un BBC es una tarea bastante complicada. Su ordenador debe ser un BBC Micro Modelo B equipado con una interface para unidad de disco. Dado que ya no se necesita la fuente de alimentación eléctrica del BBC, se deben desenchufar los siete conectores que la unen a la placa principal del ordenador. Éstos se sustituyen entonces por un cable que viene de la fuente de alimentación del Disk Pack. Luego se instala un cable plano del Disk Pack en la interface para unidad de disco del BBC. La ROM que contiene el sistema operativo MCP se enchufa entonces en un conector para ROM libre del BBC



Chris Stevens

Software gratis
El Torch Disk Pack viene con varios paquetes de software, que se ofrecen sin costo adicional alguno. Se proporciona un juego de software de oficina: Perfect Writer (un programa para tratamiento de textos), Perfect Speller (un programa verificador de ortografía), Perfect Filer (un programa de base de datos) y Perfect Calc (un programa de hoja electrónica).
Además de estos paquetes, se suministra una versión de BASIC BBC para ejecutar en el segundo procesador Z80. Las instrucciones son las mismas que en la versión en ROM del BBC con la excepción del ensamblador de 6502 incorporado del BBC. Como el segundo procesador es un Z80 y no un 6502, el BASIC BBC del Z80 tiene incluido un ensamblador Z80. La bonificación son 48 K de memoria libre, independientemente de la modalidad de visualización que se esté utilizando. El BBC Micro tiene apenas 9 K libres en algunos modos de visualización y en ningún caso tiene libres más de 28 K

Chip DFS
A la placa de circuito impreso del BBC se le han de añadir unos cuantos chips para que pueda utilizar una unidad de disco. Estos chips no vienen incluidos en el Torch Disk Pack y, por lo tanto, se han de adquirir por separado. Esta ampliación no está pensada específicamente para el Torch y contiene un chip denominado DFS que en realidad el Torch no utiliza. Sin embargo, vale la pena tenerlo, porque el DFS (*disk filing system*: sistema de archivo en disco) es el sistema operativo que utiliza Acorn. Esto significa que el Torch Disk Pack se puede emplear como un par de unidades de disco para ejecutar software en disco para el BBC Micro en el procesador 6502.
Los formatos de disco de Torch y Acorn no son compatibles, de modo que los programas MCP no pueden leer discos Acorn y viceversa. Con el Torch se suministra un programa que convierte de un formato a otro



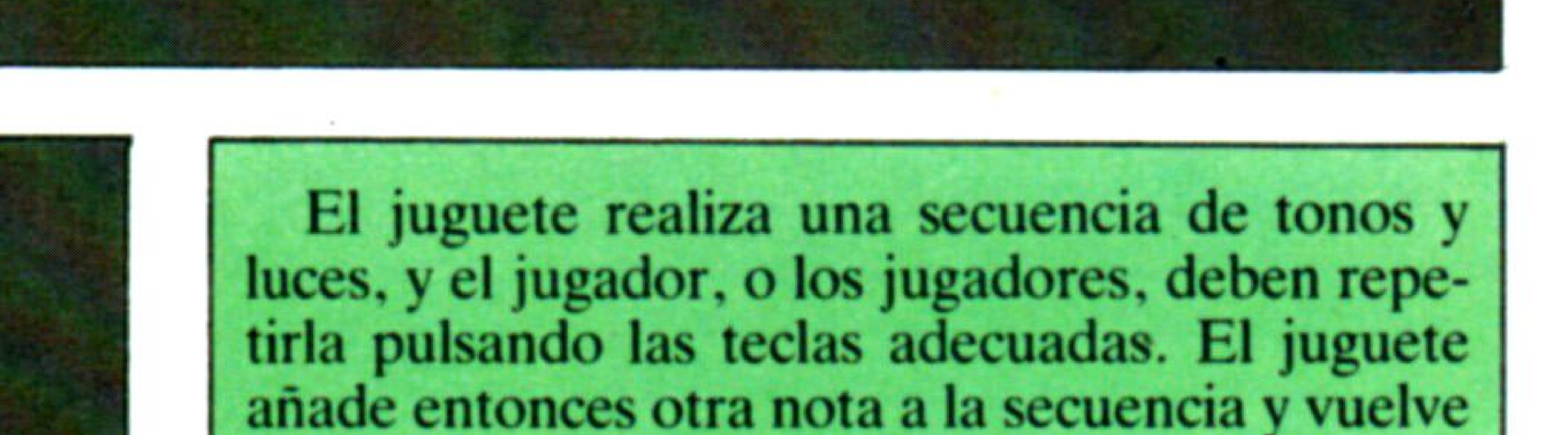
Opciones para oficina
El primer ordenador que produjo Torch fue una máquina de oficina. Torch produce, asimismo, versiones de la máquina de oficina que utiliza el sistema operativo Unix. Ésta es la serie 700 (que vemos en la fotografía). La firma tiene también en el mercado la serie 300, terminales que enlazan los ordenadores Torch en una red

Color por números

En este capítulo analizaremos un juego por ordenador aparecido hace algún tiempo, que se basa en otro juego muy conocido: el tradicional "Simón dice"

"Simón dice" es uno de los primeros juegos al que muchos de nosotros recordamos haber jugado. El que lleva el juego da instrucciones como "Simón dice colocar las manos sobre la cabeza", o bien "Simón dice pararse", etc. Los jugadores quedan descalificados si responden a una orden no comenzada con "Simón dice...".

Por sorprendente que parezca, este juego típico de fiestas o de clase se convirtió en un juego electrónico cuando se desarrollaron numerosos juguetes electrónicos que utilizaban microprocesadores para controlar una serie de botones, luces y zumbadores. En la versión electrónica, el ordenador es el líder del juego, y es obligatorio seguir todas sus instrucciones.



El juguete realiza una secuencia de tonos y luces, y el jugador, o los jugadores, deben repetirla pulsando las teclas adecuadas. El juguete añade entonces otra nota a la secuencia y vuelve a comenzar.

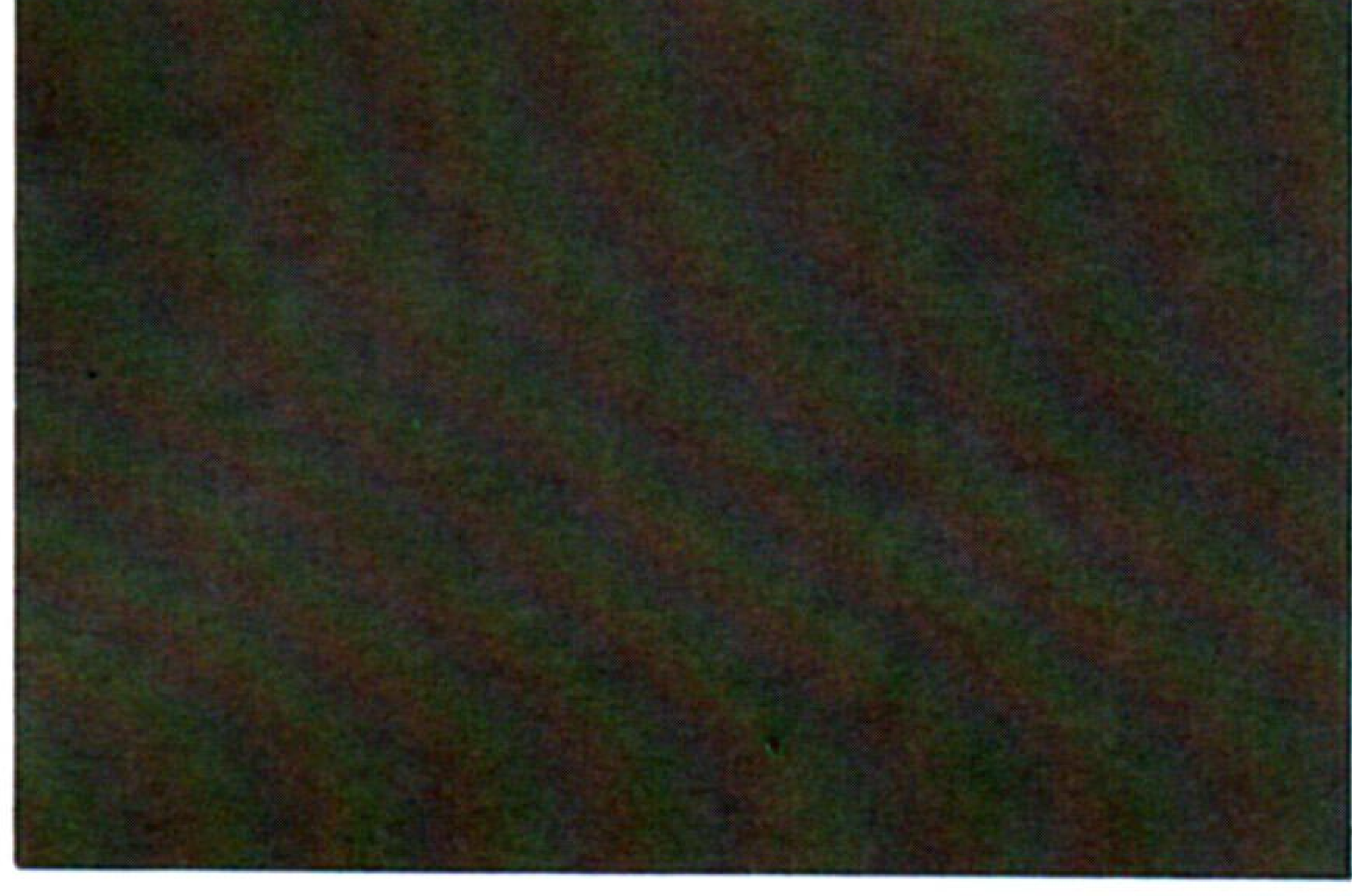
Por supuesto, si se posee un ordenador personal, la idea de una máquina dedicada exclusivamente a jugar le parecerá más bien extraña. Un juguete puede ser muy portátil, duradero y fácil de usar. Pero hasta los mejores juegos pronto resultan aburridos ¡y la capacidad del ordenador para ejecutar cientos de programas diferentes asegura que éste jamás dejará de resultar interesante!

El programa que aquí listamos se denomina "¡Imítame!" y desarrolla un juego basado en cuatro luces de colores, cada una con su propio sonido y su propia tecla en el teclado, numeradas del 1 al 4. El programa enciende una luz y luego le pide al jugador que la repita. Si lo hace bien, el programa enciende dos luces, y así sucesivamente.

Hay dos formas en las que se puede perder en este juego: tardando demasiado tiempo en responder con el siguiente botón o pulsando un botón equivocado tres veces en una partida. Para hacer que el juego resulte más difícil, esta versión también va más deprisa a medida que se va avanzando, si bien no es preciso reproducir la secuencia a la misma velocidad que el ordenador. En el juego cada secuencia tiene un máximo de 50 luces; es realmente poco probable que usted llegue a este límite; ¡la mayoría de los jugadores consiguen responder hasta una secuencia de 15 luces antes de que el juego resulte demasiado para ellos!

Es interesante observar la forma en que está estructurado el programa. Todas las instrucciones de color y sonido están agrupadas en subrutinas al final del programa del siguiente modo:

1000 Visualizar número de luz a
1500 Obtener una pulsación de tecla 1, 2, 3 o 4
2000 Hacer un ruido para el final del juego
2500 Hacer un ruido para avisar que la respuesta ha sido errónea
6000 Visualizar un mensaje en la línea 20 de la pantalla y, de ser necesario, hacer una pausa después



En un juguete exclusivo, estas subrutinas podrían corresponder a dispositivos reales de hardware. Extraerlas del cuerpo principal del programa tiene dos efectos positivos. En primer lugar, todas las complejidades de sonido y color específicas de la máquina se mantienen en un lugar, haciendo que resulte más fácil llevar el programa de una máquina a otra. En segundo lugar, sería fácil utilizar las subrutinas para un juego diferente siguiendo aproximadamente las mismas líneas. El mismo programa podría ofrecer variantes del juego, así como sucede con los juguetes exclusivos. Quizá a usted le gustaría inventarse algunas variantes propias y agregarlas al programa. Por ejemplo, éste se podría adaptar para que se visualicen los marcadores individuales para cualquier número de jugadores.

Como usuario de un ordenador, no olvide que cualquier cosa que pueda hacer un juguete exclusivo, su ordenador lo puede hacer mejor.

Kevin Jones

¡Imítame!

```
10 REM Juego Imitame!
30 LET h = 0: LET n = 0: LET w = 3
40 DIM c(4): DIM p(4): DIM a(50)
50 LET p(1) = 5: LET p(2) = 8: LET p(3) = 12: LET p(4) = 15
60 LET c(1) = 1: LET c(2) = 2: LET c(3) = 4: LET c(4) = 6
70 LET s$ = "  ": FOR i = 1 TO 5: LET s$ = s$ + s$: NEXT i
80 LET b$ = CHR$ (143): REM un bloque
100 REM *** pagina de instrucciones
110 CLS: PRINT TAB (10):"Imitame!"
120 PRINT: PRINT
130 IF n > 0 THEN PRINT: PRINT "Lo has conseguido  ";n;"  veces"
140 IF n > h THEN PRINT: PRINT "... Un nuevo record!!!": LET h = n
150 IF h > 0 THEN PRINT: PRINT "Lo mejor hasta ahora son";h;"  veces"
155 PRINT: PRINT "Intenta reproducir la secuencia de luces y sonidos del
    ordenador"
160 PRINT "pulsando las teclas del 1 al 4"
170 PRINT: PRINT "Pulsar P para jugar, S para parar"
180 LET a$ = INKEY$: IF a$ = "" THEN GO TO 180
190 IF a$ = "s" OR a$ = "S" THEN CLS: STOP
200 IF a$ <> "p" AND a$ <> "P" THEN GO TO 180
205 REM *** Nuevo juego
210 CLS: PRINT TAB (10); "Imitame!"
220 FOR a = 1 TO 4: GO SUB 1000: NEXT a
230 LET n = 0: LET m = 0: RANDOMIZE
240 REM *** Siguiente turno
250 LET n = n +1
270 LET a(n) = INT (RND*4) + 1
280 IF m = w THEN GO SUB 2000: LET m$ = "*" + STR$(w) + "respuestas
    incorrectas!": GO SUB 6000: GO TO 100
285 LET m$ = "*Ahi va...": GO SUB 6000
290 FOR i = 1 TO n
300 LET a = a(i): GO SUB 1000
310 FOR j = 1 TO 100/n: NEXT j
320 NEXT i
330 LET m$ = "Imitame...": GO SUB 6000
340 LET i = 1
350 GO SUB 1500
360 IF t = 0 THEN GO SUB 2000: LET m$ = "*Demasiado lento!": GO SUB 6000:
    GO TO 100
370 IF a <> a(i) THEN LET m = m + 1: GO SUB 2500: GO TO 280
380 LET i = i + 1: IF i < = n THEN GO TO 350
390 IF n > 50 THEN LET m$ = "*Tu ganas con 50 veces!": GO SUB 6000: GO TO
    100
400 LET m$ = "*Preparate para volver a probar": GO SUB 6000
410 GO TO 250
1000 REM *** Iluminar recuadro a
1010 INK c(a)
1015 LET p = (a-1)*8 + 2
1020 FOR l = 10 TO 14
1030 PRINT AT l,p;
1040 IF l = 12 THEN PRINT b$;b$;a;b$;b$;
1050 IF l <> 12 THEN PRINT b$;b$;b$;b$;b$;
1060 NEXT l
1070 BEEP 2/(n + 1),p(a)
1080 PRINT AT 10,p;"       "
1090 PRINT AT 11,p;"  ";b$;b$;b$;"  ";
1100 PRINT AT 12,p;"  ";b$;a;b$;"  ";
1110 PRINT AT 13,p;"  ";b$;b$;b$;"  ";
1120 PRINT AT 14,p;"       "
1130 INK 0: RETURN
1500 REM *** Leer una tecla
1510 LET t = 250
1520 LET a$ = INKEY$; IF a$ = "" THEN LET t = t-1: IF t > 0 THEN GO TO
     1520
1530 IF t = 0 THEN RETURN
1540 IF a$ <> "1" AND a$ <> "2" AND a$ <>"3" AND a$ <> "4" THEN GO
     TO 1520
1550 LET a = VAL (a$): GO SUB 1000
1560 RETURN
2000 REM *** Despedida
2010 BEEP 3,0: RETURN
2500 REM Advertencia
2510 BEEP 1,0: RETURN
6000 REM *** Imprimir m$
6010 PRINT AT 20,1;s$;AT 20,1;
6030 IF m$(1) = "*" THEN PRINT m$(2 TO): FOR z = 1 TO 200: NEXT z
6040 IF m$(1) <>"*" THEN PRINT m$
6050 RETURN
```

Complementos al BASIC

Commodore 64

Reemplazar CLS por PRINT CHR$(147).
Reemplazar LET A$ = INKEY$ por GET A$
Reemplazar RANDOMIZE por XX = RND(−TI)
Reemplazar (RND*4) por (RND(1)*4).
Reemplazar M$(1) por LEFT$(M$,1).
Reemplazar M$(2 TO) por MID$(M$,2).
Reemplazar CHR$(143) por CHR$(166).
Reemplazar PRINT AT L,C; por PRINT LEFT$(DN$,L + 1)TAB(C); (p. ej., la línea 6010 se convierte en 6010 PRINT LEFT$(DN$,21)TAB(1);S$;LEFT$(DN$,21)TAB(1);
Reemplazar DIM C(4) por DIM C$(4)
Reemplazar b$;b$;a;b$;b$; por BBZBB$; en la 1040.
Reemplazar b$;a;b$; por BZB$ en la 1100
Insertar:

```
20 VL=54296:AD=54277:SR=AD+1:WF=
   AD-1:NO=17:N1=NO:LF=AD-5:HF=LF+1
25 POKE AD,255:POKE SR,48:POKE VL,15
50 C$(1) = CHR$(31):C$(2) = CHR$(28):C$(3)
   = CHR$(30):C$(4) = CHR$(158)
60 P(1) = 51:P(2) = 34:P(3) = 64:P(4) = 38
90 DN$ = CHR$(17):FOR K = 1 TO 5:DN$ = DN$
   + DN$:NEXT K:DN$ = CHR$(19) + DN$
1010 PRINT C$(A);
1015 P = (A-1)*9 + 3:Z$ = RIGHT$(STR$(A),1)
1130 PRINT CHR$(144):RETURN
2010 SD = 15:SP = 4:N1 = 33:GOSUB
     7000:RETURN
2510 SD = 10:SP = 10:N1 = 33:GOSUB
     7000:RETURN
7000 REM *** BEEP SD,SP
7010 POKE VL,15:POKE WF,N1
7020 POKE LF,SP:POKE HF,SP: FOR DD = 1 TO
     SD*50:NEXT DD
7030 POKE HF,0:POKE LF,0:N1 = NO:RETURN
```

BBC Micro

Reemplazar AT Y,X por TAB(X,Y). Por ejemplo, la línea 6010 se transforma en:

```
6010 PRINT TAB(1,20);S$;TAB(1,20)
```

Reemplazar INKEY$ por INKEY$(0). Insertar:

```
20 MODE 2
25 COLOUR 135:CLS
60 C(1) = 1:C(2) = 2:C(3) = 4:C(4) = 5
80 B$ = CHR$(35)
1010 COLOUR C(A)
1015 P = (A-1)*4
1070 SOUND 1,-10,P(A),40/(N + 1):FOR
     DE = 1 TO 2000/N:NEXT
1130 COLOUR 0:RETURN
2010 SOUND 1,-15,2,40
2510 SOUND 1,-15,40,40
```

Plan de acción

En esta ocasión estudiaremos el diseño de un programa y consideraremos las preguntas que se deben formular antes de escribir un código

El diseño de programas está considerado por quienes están implicados en el mismo (el diseñador-programador y el usuario) como un ejercicio formal y excelente de una aplicación de "resolución de problemas". Lamentablemente, los problemas a resolver siempre se asume que son de tipo técnico, cómo formatear la pantalla, cómo lograr que un bucle sea más rápido, dónde colocar todo en la RAM, etc., mientras que los problemas reales están presentes desde el comienzo del proyecto y por lo general se crean en el primer encuentro del usuario y el "experto". Es raro que los usuarios sean muy conscientes de la verdadera naturaleza de sus problemas (esperan que el experto les diga cuál es el problema y cómo resolverlo) y los expertos muy a menudo piensan que conocen el problema y la solución antes de que el usuario empiece siquiera a enunciarlo. El resultado es una mala comunicación inicial, que lleva a una descripción incompleta del problema y las necesidades del usuario. Trabajar partiendo de estos principios da como resultado la producción de un sistema insatisfactorio que el usuario se verá forzado a aceptar.

En los proyectos de informática personal el programador por lo general es también el diseñador (o "analista de sistemas") y el consumidor. Esto debería significar que los problemas de comunicación se redujeran considerablemente. No obstante, como un usuario-diseñador combinado, usted siempre debería hacer el esfuerzo de explicarse a sí mismo los problemas, las soluciones y las necesidades con la misma claridad que si estuviera hablando con otra persona.

Consideremos el caso de un usuario imaginario y su problema: es un modelista de aviones a escala aficionado que también posee un microordenador basado en cassette. Desea almacenar descripciones bastante detalladas de los materiales que utiliza para cada uno de los modelos que construye de modo que cuando trabaje en los últimos modelos pueda buscar en sus registros los empleos anteriores de esta clase de pegamento o de este tipo de juntura. Por consiguiente, lo que el diseñador debe obtener del usuario es una clara enunciación de lo siguiente:

■ La función del programa. Esto puede empezar como una vaga enunciación de intenciones, tal como "Deberá almacenar los registros de mis modelos", pero debe irse afinando mediante la persuasión e interrogación del diseñador en algo más parecido a una especificación de requerimientos, como "Debería almacenar mis descripciones del modelo y su construcción y materiales, entrando toda la información por el teclado, y visualizándola cuando yo entre el nombre del modelo o algún aspecto de su construcción". Esto enuncia las necesidades del usuario con bastante más claridad y señala algunas de las tareas específicas de programación implicadas (almacenamiento, búsqueda, indexación, recuperación, etc).

■ Cómo se utilizará el programa. Algunos de los detalles físicos de la utilización típica pueden quedar claros a partir de la descripción de la función, pero puede que no estén completos. Por ejemplo, tal vez el usuario no desee que los detalles del modelo se visualicen en la pantalla porque él trabaja en un cobertizo donde no tiene un aparato de televisión. En este caso, tal vez lo más indicado fuera una "copia en papel" de los detalles seleccionados.

■ Qué aspecto tendrá: los formatos de entrada y salida. El programador profesional con frecuencia empleará diagramas preimpresos que representen la pantalla para trazar cada una de las visualizaciones que el usuario verá durante las fases de entrada-salida. Estas técnicas no suelen ser necesarias para el uso personal, si bien los gráficos en alta resolución pueden representar la excepción a esta regla. Los formatos de pantalla son un aspecto muy importante de la *interface para el usuario* (lo que el usuario "ve" del programa) y merecen la misma atención y el mismo análisis que a veces se les dedica a aspectos obviamente más ergonómicos de la informática, como la disposición de teclados y pantallas, la altura de la mesa, los niveles de iluminación, etc.

■ Cómo se habrá de organizar: formatos de archivo y programa. Tal vez el usuario piense que necesita almacenar al menos 100 descripciones de aviones, y que cualquier cantidad inferior sería insuficiente. Por el contrario, quizá nunca llegue a construir más de media docena de modelos estándar. Las dimensiones del archivo de datos tienen serias implicaciones respecto a su formato y métodos de acceso. Una exploración secuencial a través de seis descripciones de modelos en cassette que tarde, supongamos, cinco minutos, podría ser bastante aceptable para el usuario, mientras que esperar a que se buscaran 100 ya no tendría sentido. Una solución podría ser colocar el programa y el archivo de índice de descripciones en una cinta, y las descripciones propiamente dichas en otras 20 cintas clasificadas por tipo de avión, por ejemplo.

Las dimensiones del propio programa también pueden constituir un problema: si la sección de entrada de texto exige un editor de textos complejo, si el programa está lleno de menús y muchos mensajes significativos, si las secciones de tratamiento de archivos emplean complicadas rutinas de búsqueda e indexación, entonces el programa tal vez se tendrá que dividir en varios programas separados con el fin de que quepa en la RAM disponible.

■ Qué deberá hacer: procedimientos y cálculos especiales. En el ejemplo de los aviones a escala, es poco probable que surjan, pero a menudo sí apare-



cen cuando se consideran otros programas. Puede que existan 20 formas equivalentes y perfectamente buenas de pasar por un proceso determinado, pero cabe en lo posible que el usuario insistiera en una y sólo una de ellas.

Realizar esto mal hace que el usuario quede de inmediato insatisfecho con el programa. El diseñador se puede sentir tentado a no usar el método preferido por el usuario y sí otros métodos más eficaces, ¡pero cualquier posible ventaja desaparece enseguida cuando el usuario no quiere luego utilizar el programa! Descubrir los procedimientos del usuario puede ser muy útil cuando se trata de diseñar cálculos. ¿Por qué inventar una fórmula para calcular las cargas alares, por ejemplo, si uno simplemente le puede preguntar al que construye el modelo cómo lo hace él?

Teniendo toda esta información apuntada, ya se puede comenzar el trabajo de convertir estas especificaciones en un programa. Un enfoque útil es el de diseñar primero el diálogo usuario-programa, luego los archivos de datos y luego los procesos que controlan todo. La palabra "diálogo" se toma en el sentido de la comunicación bidireccional de información que se produce entre el usuario y el programa. Ésta no consiste sencillamente en la entrada de los detalles de los modelos de avión y su subsiguiente visualización, sino que también incluye todas las preguntas, mensajes y menús que el programa produce y todas las entradas, órdenes o selecciones que entra el usuario. Asimismo, es importante determinar el tipo de diálogo en esta etapa. Para el programa de los aviones podría ser adecuada una elección entre menú y acciones mediante órdenes. Las decisiones tomadas aquí tendrán un considerable efecto en la estructura del programa general. El contenido y el formato del diálogo se debe considerar en detalle, pero la recompensa a este esfuerzo es que todos los datos manipulados por el programa son especificados en esta fase. Ello significa que se puede calcular el tamaño de memoria requerido para mensajes de error y textos de preguntas, y que los archivos se pueden diseñar ahora.

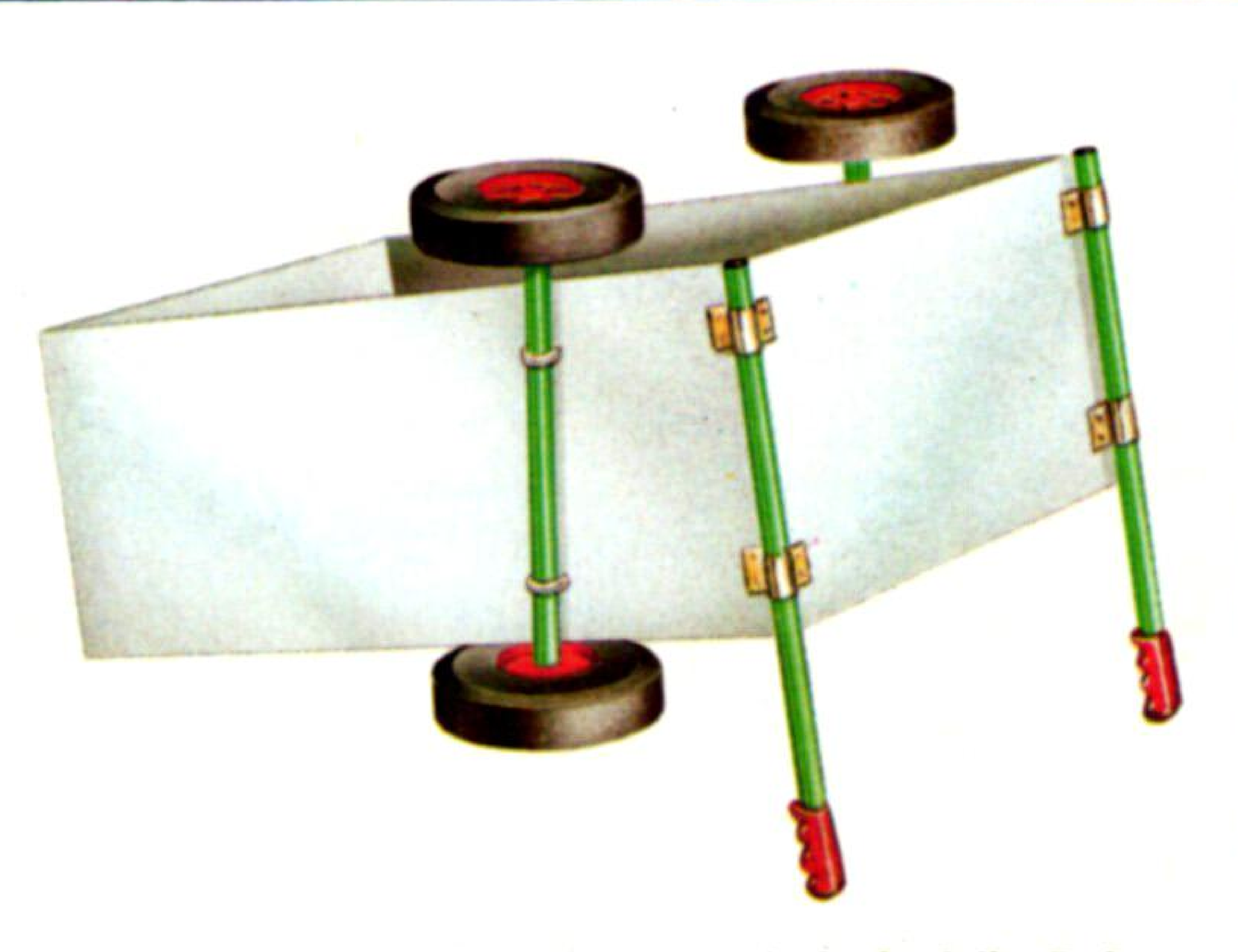
Para el programa de los aviones a escala, donde los archivos contendrán grandes bloques de texto y serán muy extensos, la mejor solución sería la de dividir el archivo en varias cintas de modo que resulte más fácil buscar en cada una de ellas. Si valiera la pena (depende del volumen), los datos se podrían comprimir mediante un algoritmo de codificación antes de escribirlos en cinta, y después decodificarlos durante la lectura.

Ya en esta fase, las funciones necesarias serán evidentes. Habrá rutinas que permitan añadir y editar textos de datos, para archivar los textos recién introducidos (éstas actualizarán todos los índices utilizados por el sistema), para aceptar nombres de componentes, para buscar y visualizar descripciones, etc. Todo ello se le debe presentar al usuario en forma de opciones, y el programa debe ser capaz de detectar y tratar datos que no sean válidos.

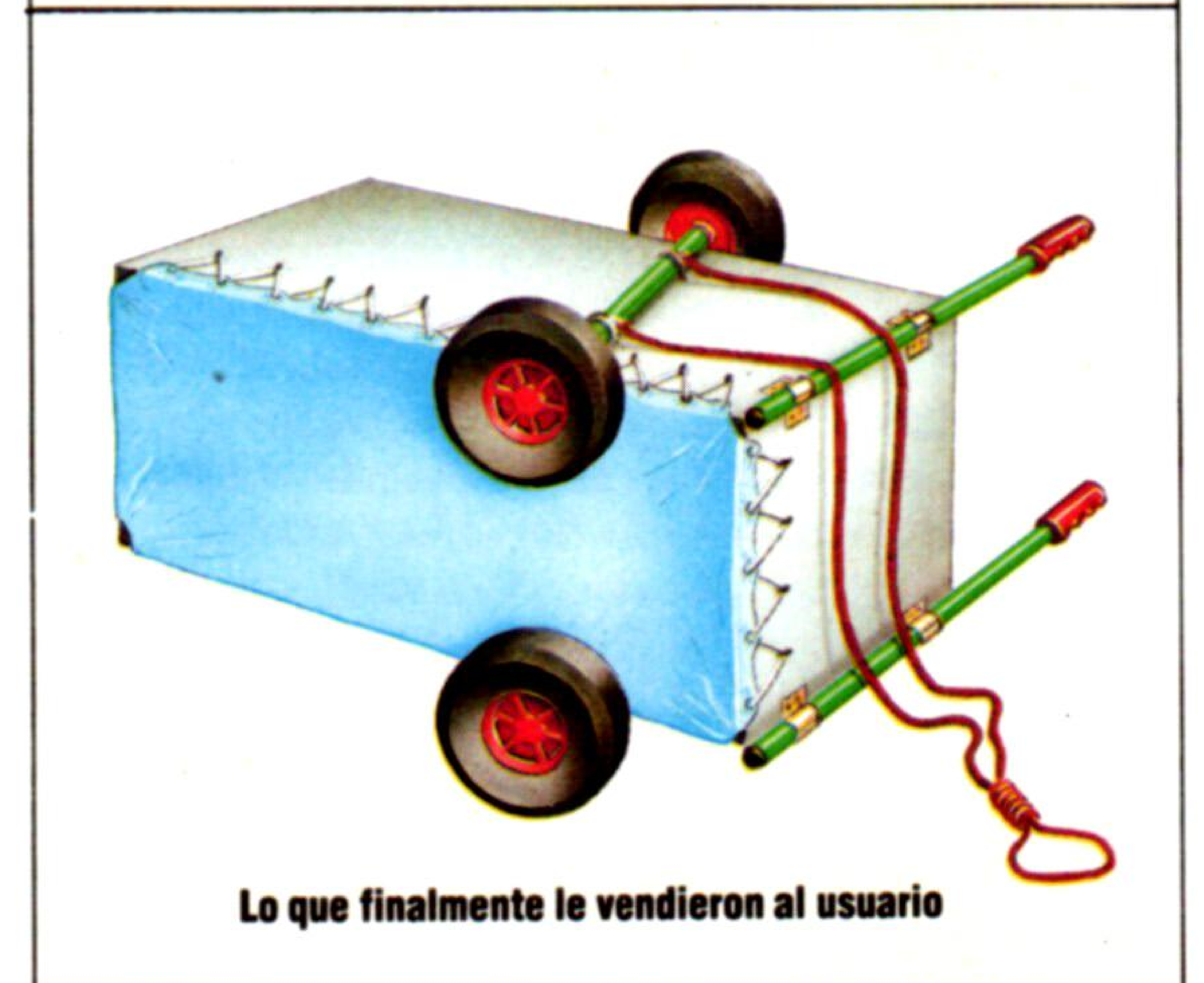
En esta etapa es aconsejable que el usuario verifique concienzudamente el diseño para asegurar que hace lo que debería hacer. Si todo está bien, entonces se puede codificar el programa. Por supuesto, es más fácil decirlo que realizarlo, y el acto de convertir el diseño en un programa que funcione eficazmente muy bien podría poner en evidencia otros problemas.



Lo que el diseñador pensó que necesitaba el usuario



Lo que el programador pensó que quería decir el diseñador



Lo que finalmente le vendieron al usuario



Lo que en realidad deseaba el usuario

Steve Cross

Describir, definir, diseñar
Si el equipo de desarrollo de software típico (o sea, usuario, diseñador y programador) se hubiera dedicado a resolver el problema de desplazar cargas pesadas por los jardines, esto es lo que habría hecho. La mala comunicación (entre experto y no experto, y también entre expertos) sigue siendo un importante problema con que se enfrentan todos los equipos de diseño

Los fantasmas del ordenador

"Atic Atac" une a los elementos estratégicos propios de los juegos de aventuras la rápida acción de los juegos recreativos

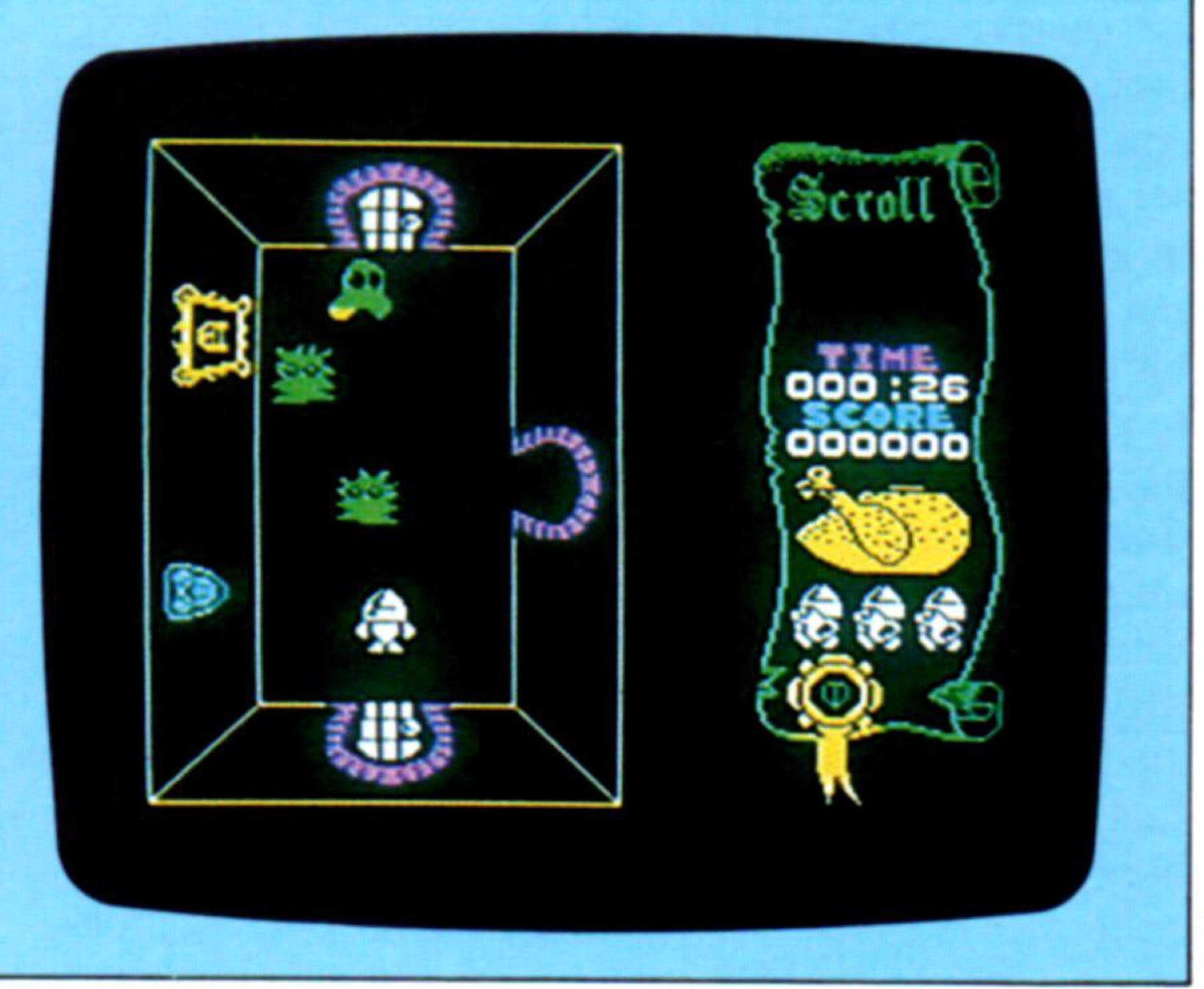
Atic Atac es uno de los juegos de esa rara casta que consiguen combinar la emoción de la acción de los recreativos con la complejidad de los juegos de aventuras. Respondiendo al estilo clásico de aventuras, el juego se desarrolla en las numerosas dependencias de un castillo habitado por fantasmas. El jugador está atrapado dentro del castillo y sólo puede escapar encontrando la llave de oro que abre las puertas principales. Su vida se ve amenazada por una serie de criaturas perversas: arañas, profanadores de tumbas, brujas, demonios y monstruos hambrientos, todos ellos están allí, por no hablar de Drácula, el monstruo de Frankenstein, la Momia y gran cantidad de murciélagos. Es algo así como hallarse de pronto protagonizando una superpoblada película del más puro horror. Las puertas falsas y los pasajes ocultos abundan y hay objetos útiles o valiosos que el jugador puede recoger.

Hasta ahora todo suena como un típico juego de aventuras, pero está ilustrado con espléndidos gráficos animados; y aquí es donde comienza el aspecto recreativo de *Atic Atac*. Al jugador se le presenta una panorámica en color y tridimensional de cada cuarto o calabozo, donde pueden verse las diversas puertas que dan al norte, al sur, al este o al oeste. También hay adornos que incluyen armaduras, estanterías con libros, relojes del abuelo y cuadros. Vestido con el disfraz elegido de Caballero, Mago o Siervo, el jugador se desplaza por las habitaciones protegido con el arma apropiada para su personaje. Los numerosos monstruos no cesan de aparecer, envueltos en nubes de humo, y su solo contacto hace que el protagonista se debilite muchísimo. Afortunadamente, con el uso de las armas, hacha, espada o conjuro, puede repelerlos.

Habitantes del castillo

Estas fotografías de pantalla muestran dos etapas del *Atic Atac*, un juego de aventuras que se desarrolla al frenético ritmo de un juego recreativo. Una sucesión de monstruos y otras sorpresas se le aparecen al jugador mientras recorre corredores y pasadizos secretos en busca de una llave de oro





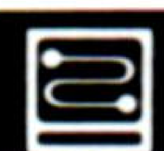
Liz Heaney

El movimiento se controla mediante una palanca de mando Kempston o de cursor, o con el teclado. Lamentablemente, al igual que muchos juegos, *Atic Atac* utiliza para el movimiento las teclas Q, W, E y R, y como están dispuestas sobre el teclado en línea recta, el juego resulta difícil de jugar.

El juego se podría convertir fácilmente en un simple juego vertiginoso si no fuera por el hecho de que algunos de los objetos, por no hablar de la propia llave de oro, son muy útiles. Algunas puertas de colores sólo se abrirán si se posee la llave del mismo color y ciertos objetos protegen contra determinadas criaturas. En un nivel más básico se necesita hallar comida, como indica tan a las claras el pollo que se desintegra rápidamente a la derecha de la pantalla. Si no se lo come, poco a poco la saludable ave se transforma en un montoncito de huesos, lo que significa la muerte del usuario por inanición.

Al principio, *Atic Atac* se tiende a considerar como un simple juego recreativo muy frustrante. Una vez que comience a dominar las técnicas para enfrentarse a los numerosos monstruos, empezará a apreciar el aspecto de aventura del mismo, mientras busca por el castillo los diversos artefactos y tesoros. Pero se lo advertimos: éste es un juego que exige muchas horas de esfuerzo para hallar las llaves con las que poder abrir las puertas.

Atic Atac: Para el Spectrum de 48 K
Editado por: Ashby Computers and Graphics Ltd., Ashby de la Zouch, Leicestershire LE6 5JU
Autores: Ultimate, Play The Game
Palancas de mando: Kempston, y de cursor
Formato: Cassette



Siluetas rutinarias

En este capítulo examinamos una rutina para crear y mover un dibujo empleando el BBC Micro

Muchas son las formas y figuras que pueden adoptar los sprites, pero todas tienen en común el reticulado o cuadrícula. No existe ninguna restricción al elegir el tamaño de la cuadrícula, pero resulta práctico que se componga de un número entero de bytes (ocho, dieciséis, veinticuatro bits, etc.) formando un rectángulo o cuadrado. La rutina que vamos a proporcionarle y comentar se vale de una cuadrícula de 24 pixels de largo por 21 de ancho, que necesita, por tanto, 63 bytes de memoria para representar el sprite. Si trazamos el dibujo que deseamos sobre este rectángulo cuadriculado, sólo queda traducirlo a binario para definir un sprite. Todo pixel "encendido" corresponde a un 1 y los "apagados" equivalen a 0. Una vez obtenidos los valores binarios correspondientes, debemos convertirlos a decimal, o a hexa, y a continuación colocarlos en alguna parte de la memoria. La ilustración que el lector puede ver en la parte inferior de esta página muestra el dibujo del sprite que hemos escogido.

Los 63 números que componen el sprite pueden definirse usando la sentencia DATA y ser leídos (READ) para su tratamiento en BASIC. La instrucción READ se encarga de colocarlos en algún área de la memoria. Esta área puede estar en cualquier zona de la RAM con tal de que no se escriba nada encima durante la ejecución del programa. El sitio más obvio para almacenarlos sería la parte superior de la memoria destinada al programa en BASIC. La dirección superior de esta área está contenida en HIMEM (HI*gh*: alta). Si queremos que ningún otro dato venga a superponerse en ella, bajaremos un poco HIMEM. Esto es lo que hacen las líneas 220 y 230 del programa, al tiempo que fijan la dirección del primero de los bytes de DATA, que llamamos SPRDAT (datos del sprite). Por último, las líneas 1740 a 1770 "leen" (READ) los datos y los colocan en los 63 bytes consecutivos que empiezan en la posición SPRDAT.

La rutina en el código máquina

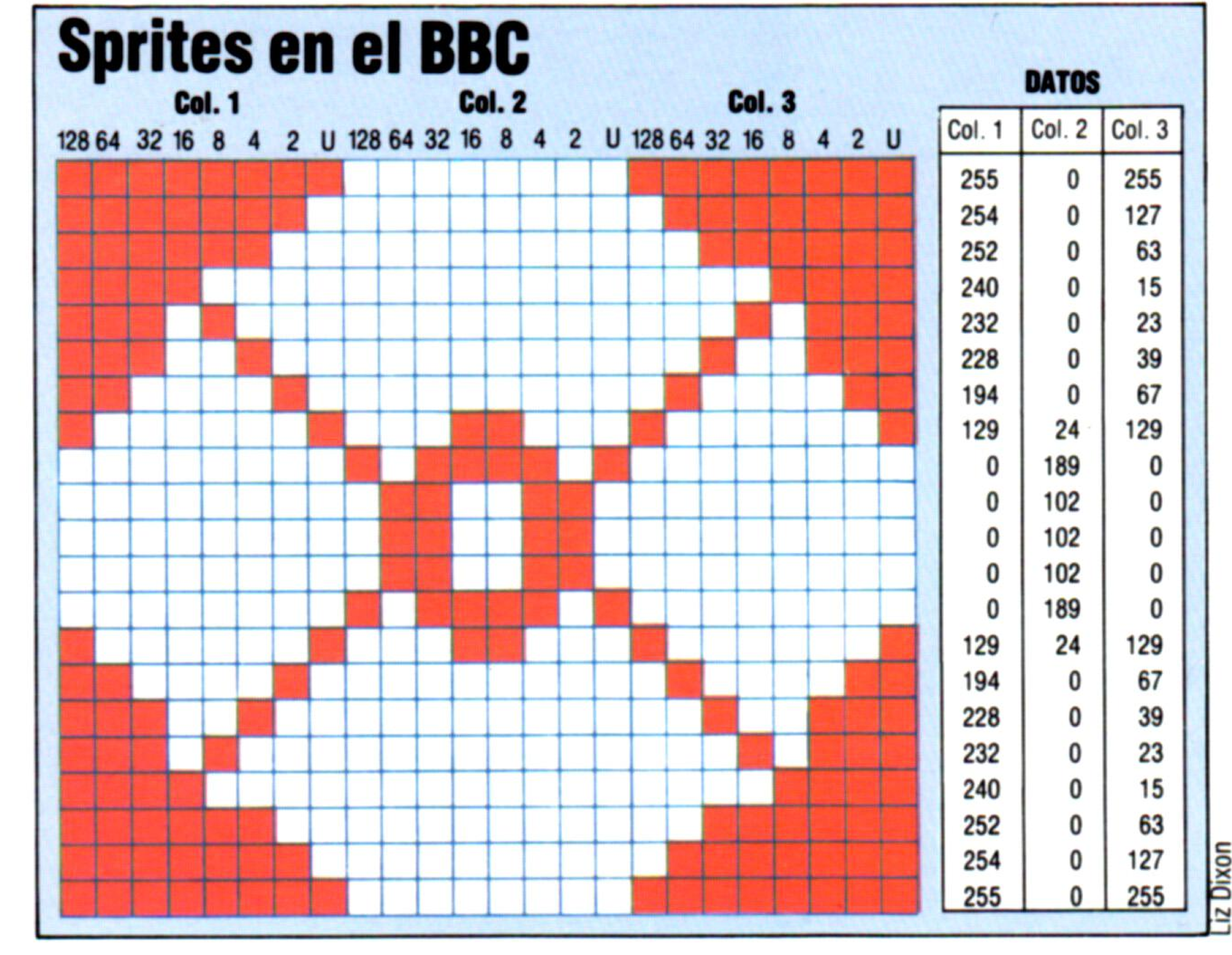
Lo más importante que encomendaremos al código máquina es el análisis del dibujo que define el sprite escogido y la ejecución de las operaciones necesarias para convertir los datos en su correspondiente visualización sobre la pantalla. Esto se consigue examinando cada bit de los referidos 63 bytes, uno a uno, para trazar un punto si el bit está a 1, o bien dejar un espacio si está a 0. La manera más fácil de analizar cada bit de un byte determinado es quizá empleando una de las instrucciones de rotación. La línea 1100 se sirve de ROL (RO*tate* L*eft*: rotar a la izquierda) referida a un byte determinado. Ya sabemos que esta instrucción desplaza cada bit del byte un lugar a la izquierda. Como el valor previo del flag de arrastre se coloca en el extremo derecho del byte, de la misma manera el bit que "sobra" por el otro extremo izquierdo se coloca en el flag. Con lo cual es posible examinar cada bit a su paso por el flag o indicador. Siempre que se tenga la precaución de no alterar el valor del flag durante el proceso, el byte original se recupera intacto después de nueve rotaciones.

Veamos ahora más gráficamente lo que acabamos de exponer:

	Flag	Byte
Byte y flag de partida	C	1 1 0 1 1 1 0 0
Después del 1.er ROL	1	1 0 1 1 1 0 0 C
Después del 2.º ROL	1	0 1 1 1 0 0 C 1
Después del 3.er ROL	0	1 1 1 0 0 C 1 1
Después del 4.º ROL	1	1 1 0 0 C 1 1 0
Después del 5.º ROL	1	1 0 0 C 1 1 0 1
Después del 6.º ROL	1	0 0 C 1 1 0 1 1
Después del 7.º ROL	0	0 C 1 1 0 1 1 1
Después del 8.º ROL	0	C 1 1 0 1 1 1 0
Después del 9.º ROL	C	1 1 0 1 1 1 0 0

Del ejemplo expuesto arriba ya habrá deducido usted que el contenido inicial o final del flag nos trae completamente sin cuidado (por eso lo hemos señalado con una C en ambos casos). En la línea

Dimensionar el sprite
El sprite mide 24 × 21 pixels, por lo que se representa en 63 bytes, ya que a cada pixel corresponde un bit en el método acostumbrado de alta resolución. La variable *logcol* se encarga del color del sprite entero, y el tamaño del sprite viene dado por dos factores, XSCALE e YSCALE. Todos los números deben ser pares



DATOS

Col. 1	Col. 2	Col. 3
255	0	255
254	0	127
252	0	63
240	0	15
232	0	23
228	0	39
194	0	67
129	24	129
0	189	0
0	102	0
0	102	0
0	102	0
0	189	0
129	24	129
194	0	67
228	0	39
232	0	23
240	0	15
252	0	63
254	0	127
255	0	255

1100 del programa se emplea ROL en el modo de direccionamiento indexado absoluto, lo que permite el acceso a los 63 bytes y su análisis como en el byte del ejemplo.

Ya tenemos cada bit aislado y examinado para saber si corresponde a punto o espacio: ahora falta su trazado sobre la pantalla. El BBC Micro puede realizar esto de dos maneras. La primera consiste en colocar (POKE) los valores directamente en el área para visualización de la memoria. Lo cual no es tan fácil como parece en este ordenador, donde se nos presentan dos problemas principales. Uno de ellos surge de las diferencias que existen entre las áreas de memoria reservadas a la pantalla en el modelo A y en el B y también entre distintos modos. Otro es que la relación matemática entre los pixels y los bits de la memoria de pantalla es de una considerable complejidad. Por ejemplo, en el modo 2 cada byte de la memoria controla tan sólo dos pixels de la pantalla, pues este modo dispone de 16 colores y cada pixel necesita cuatro bits para la definición de su color. Ahora bien, en el modo 0, que es de dos colores, cada pixel no necesita más que un bit para su definición. Desde luego, es factible escribir una rutina en lenguaje máquina para tratar nuestro sprite en el modo 2, pero sólo funcionará en éste y en ningún modo más.

Por suerte, existe una segunda manera de trazar pixels en la pantalla. Cuando nos servimos de los gráficos en BASIC, el sistema operativo del BBC realiza el tipo de operaciones que nosotros estamos buscando. Pues bien, es posible el acceso a esta rutina del sistema operativo. Es más, el fragmento de programa que escribamos para llamar a esta rutina funciona en todos los modos. Es parecido a la instrucción VDU en el BASIC del BBC. Así, en el caso de que nos propongamos trazar un punto en la pantalla, se puede escribir:

```
VDU25,68,300;700;
```

Los últimos números, 300 y 700, son las coordenadas *x* e *y* respectivamente. Una instrucción gemela a ésta en código máquina es OSWRCH, con la que realizamos una llamada al sistema operativo. Tal llamada se repite colocando cada vez un número en el acumulador. Dado que el acumulador tiene cabida para un solo byte, las coordenadas deben ser partidas en un byte *lo* y un byte *hi*, así:

```
VDU25,68,44,1,188,2
```

Para realizar esto en lenguaje máquina, OSWRCH debe ser llamada seis veces. El vector de la dirección inicial de OSWRCH está en la posición &FFEE, y se accede a la rutina por medio de JSR &FFEE. Cualquier orden VDU se puede ejecutar de esta manera, y esta rutina que estudiamos utiliza llamadas a OSWRCH en varias ocasiones. Observe que, si bien OSWRCH no afecta a los valores de los registros X, Y y A, produce una alteración en el contenido del flag de arrastre. Por lo tanto, en el caso de que el valor del flag deba ser conservado, será necesario almacenar éste en algún sitio antes de llamar a OSWRCH. Es lo que sucede con la rutina ROL. La manera más fácil de hacerlo será trasladando el registro indicador de estado a la pila mediante PHP (véase p. 737) antes de la llamada, para recobrarlo, una vez acabada OSWRCH, mediante PLP.

Veamos ahora la estructuración de la rutina en código máquina. El análisis de los datos del sprite y trazado en pantalla se realiza con la subrutina SPRPLT (PLoT:trazar) que comienza en la línea 890. Lo primero que hace la rutina es plantearse el método de trazado como una operación OR exclusivo (operación XOR). En BASIC la orden similar es GCOL para el BBC. Para borrar los puntos así trazados basta con volver a escribir sobre ellos los mismos puntos. Cualquier dato bajo el sprite quedará intacto. Al comienzo del fragmento en código máquina se realiza un movimiento absoluto para posicionar la esquina superior izquierda del sprite. Cada fila es analizada ahora tomando tres bytes de los datos del sprite y haciéndolos rotar según explicamos anteriormente.

Un trazado o un movimiento relativo se realiza escogiendo una distancia que proporciona el factor horizontal de escala, XSCALE, dependiendo del valor del bit en curso contenido en el flag de arrastre, perteneciente a uno de los bytes del sprite. Al final de cada fila de tres bytes se vuelve a realizar un movimiento absoluto hacia el mismo punto de la abscisa *x* correspondiente a la esquina extrema izquierda del sprite y hacia otro punto de la ordenada *y* que determina el factor de escala vertical YSCALE. Esto se repite hasta concluir el análisis de los 63 bytes.

En dos ocasiones se usa la subrutina SPRPLT. En la primera el objetivo es borrar el sprite previo mediante un nuevo trazado sobre el lugar que ocupaba. En la segunda, se emplea para trazar el nuevo sprite. Una vez que éste ha sido trazado, sus coordenadas se transfieren a OLDX y OLDY, los cuales quedan de esta forma preparados para un uso ulterior de la rutina.

Empleo de la rutina en código máquina desde el BASIC

Los programadores en BASIC pueden usar fácilmente esta rutina, sin que necesariamente tengan que entenderla. Pasos que deben seguir:

1) Diseñar el sprite y colocar los datos en el área de memoria como se muestra en el programa;
2) Establecer el modo de visualización que se desea usar;
3) Dar los valores de XSCALE e YSCALE conforme se indica en la línea 1870;
4) Establecer el color lógico del sprite según la línea 1890;
5) Establecer las coordenadas de la posición en que se desea que aparezca el sprite y emplear el procedimiento indicado en las líneas 2010 a 2060 para convertir las coordenadas absolutas en la forma "byte *lo*, byte *hi*";
6) CALL SPRITE (llamar al sprite).

El listado en BASIC puede, como aquí, incluir esta rutina en lenguaje máquina. El listado en assembler se puede omitir cambiando la línea 260 así:

```
FOR opt% = 0TO2STEP2
```

O, si prefiere conservar la rutina una vez ensamblada (o sea, tras ser ejecutada), utilizará ■ *SAVE, tomando buena nota de las direcciones de inicio y final del código al visualizar el listado en assembly.



Un sprite BBC

```
10 REM **** SPRITES BBC ****
20 :
30 REM ** PREPARACION VARIABLES P. 0 **
40 TYPE      =&70
50 OLDXLO    =&71:?OLDXLO=0
60 OLDXHI    =&72:?OLDXHI=0
70 OLDYLO    =&73:?OLDYLO=0
80 OLDYHI    =&74:?OLDYHI=0
90 NEWXLO    =&75
100 NEWXHI   =&76
110 NEWYLO   =&77
120 NEWYHI   =&78
130 XSCALE   =&79
140 YSCALE   =&7A
150 1 ogcol  =&7B
160 ROW      =&7C
170 YTMPLO   =&7D
180 YTMPHI   =&7E
190 XTMPLO   =&7F
200 XTMPHI   =&80
210 :
220 HIMEM=HIMEM-150
230 SPRDAT   =HIMEM+
240 OSWRCH   =&FFEE
250 DIM MC% &01FF
260 FOR op t% =OTO 3 STEP 3
270    P%=MC%
280    [
290    OPT op t%
300    \**** MUEVE A LOS ANTIGUOS X,Y ****
310    \
320    .SPRITE     LDA   #25
330                JSR   OSWRCH
340                LDA   #68
350                JSR   OSWRCH
360                LDA   OLDXLO
370                STA   XTMPLO
380                JSR   OSWRCH
390                LDA   OLDXHI
400                STA   XTMPHI
410                JSR   OSWRCH
420                LDA   OLDYLO
430                STA   YTMPLO
440                JSR   OSWRCH
450                LDA   OLDYHI
460                STA   YTMPHI
470                JSR   OSWRCH
480    \
490    \**** BORRADO ANTIGUO SPRITE ****
500    \
510                JSR SPRPLOT
520    \
530    \****MUEVE A LOS NUEVOS X,Y ****
540    .NEWMOV     LDA   #25
550                JSR   OSWRCH
560                LDA   #68
570                JSR   OSWRCH
580                LDA   NEWXLO
590                STA   XTMPLO
600                JSR   OSWRCH
610                LDA   NEWXHI
620                STA   XTMPHI
630                JSR   OSWRCH
640                LDA   NEWYLO
650                STA   YTMPLO
660                JSR   OSWRCH
670                LDA   NEWYHI
680                STA   YTMPHI
690                JSR   OSWRCH
700    \
710    \**** TRAZADO NUEVO SPRITE ****
720    \
730                JSR SPRPLOT
740    \
750    \**** TRASLADO DE LOS NUEVOS X,Y A LOS ANTIGUOS X,Y ****
760                LDA   NEWXLO
770                STA   OLDXLO
780                LDA   NEWXHI
790                STA   OLDXHI
800                LDA   NEWYLO
810                STA   OLDYLO
820                LDA   NEWYHI
830                STA   OLDYHI
840    \
850    **** VUELTA AL BASIC ****
860    \
870                RTS
880    \
890    \**** SUBRUTINA TRAZADO SPRITE ****
900    \
910    \** ESTABLECER TRAZADO CON OR EXCLUSIVO **
920    .SPRPLOT    LDA   #18
930                JSR   OSWRCH
940                LDA   #3
950                JSR   OSWRCH
960                LDA   1 ogcol
970                JSR   OSWRCH
980    \
990    \
1000   \** INICIALIZACION CONTADORES **
1010   \** X CUENTA BYTES, Y CUENTA BITS **
1020   \** ROW CUENTA FILAS DE 3 BYTES **
1030               LDX   #&00
1040   .NEWROW     LDA   #&00
1050               STA   ROW
1060
1070   .BYTE       LDY   #&09
1080   .BYT        LDA   #65
1090               STA   TYPE
1100               ROL   SPRDAT.X
1110               PHP         \STORE CARRY ON STACK
1120               BCS   DOPLOT
1130               LDA   #64
1140               STA   TYPE
1150   \** INSTRUCCION DE TRAZADO VDU **
1160   .DOPLOT     LDA   #25
1170               JSR   OSWRCH
1180               LDA   TYPE
1190               JSR   OSWRCH
1200               LDA   XSCALE
1210               JSR   OSWRCH
1220               LDA   #&00
1230               JSR   OSWRCH
1240               JSR   OSWRCH
1250               JSR   OSW..CH
1260   \** FIN DE INSTRUCCION TRAZADO VDU **
1270               PLP         \RETRIEVE CARRY
1280               DEY
1290               BNE BIT
1300   \** SI ACABO EL BYTE **
1310               INX
1320               CPX   #63
1330               BEQ   FINIS
1340   \** VER SI FINAL DE FILA **
1350               INC   ROW
1360               LDA   ROW
1370               CMP   #3
1380               BNE   BYTE
1390   \** SI FINAL FILA RESTAR YSCALE DE Y **
1400   \
1410               LDA   YTMPLO
1420               SEC
1430               SBC   YSCALE
1440               STA   YTMPLO
1450               BCS   NOSUB
1460               DEC   YTMPHI
1470 , \** MOV. ABSOLUTO PARA COMIENZO FILA SIGU. **
1480   \
1490   .NOSUB      LDA   #25
1500               JSR   OSWRCH
1510               LDA   #68
1520               JSR   OSWRCH
1530               LDA   XTMPLO
1540               JSR   OSWRCH
1550               LDA   XTMPHI
1560               JSR   OSWRCH
1570               LDA   YTMPLO
1580               JSR   OSWRCH
1590               LDA   YTMPHI
1600               JSR   OSWRCH
1610   \
1620   \** FILA SIGUIENTE **
1630               JMP   NEWROW
1640   \
1650   \** FIN DE SUBRUTINA **
1660   .FINIS RTS
1670   \
1680 ]
1690 NEXT
1700 :
1710 REM **** AQUI COMIENZA PROGRAMA EN BASIC ****
1720 :
1730 REM ** LEER DATOS DEL SPRITE **
1740 FOR address = SPRDAT TO SPRDAT+62
1750    READ data:?address = data
1760 NEXT address
1770 :
1780 REM **** ESTABL. PARAMETROS EN LENG. MAQU. ****
1790 :
1800 MODE1
1810 GCOL0,129
1820 CLG
1860 ?XSCALE = 4:?YSCALE = 4
1870 ?logcal = 1
1880 :
1890 X = 700:Y = 800
1900 PROCCOORDS (X,Y)
1910 CALL SPRITE
1920 :
1930 REM **** ESPERA DE TECLAS CURSOR ****
1940 FOR S = 0 TO 1 STEP 0
1950 PROCKEYS
1960 PROCCOORDS (X,Y)
1970 CALL SPRITE
1980 NEXT S
1990 END
2000 :
2010 DEF PROCCOORDS (X,Y)
2020 XH = X DIV 256:XL = X MOD 256
2030 YH = Y DIV 256:YL = Y MOD 256
2040 ?NEWXLO = XL:?NEWXHI = XH
2050 ?NEWYLO = YL:?NEWYHI = YH
2060 ENDPROC
2070 REM **** EXPLORACION DE TECLADO ****
2080 DEF PROCKEYS
2090 LOCAL LT,ZZ:LT = 2
2100 FOR ZZ = 0 TO 1 STEP 0
2110 IF INKEY(-58) THEN Y = Y+50:ZZ = LT
2120 IF INKEY(-42) THEN Y = Y-50:ZZ = LT
2130 IF INKEY(-26) THEN X = X-50:ZZ = LT
2140 IF INKEY (-122) THEN X = X+50:ZZ = LT
2150 NEXT ZZ
2160 ENDPROC
2170 REM **** DATOS DEL SPRITE ****
2180 DATA 255,0,255,254,0,127,252,0,63
2190 DATA 240,0,15,232,0,23,228,0,39
2200 DATA 194,0,67,129,24,129,0,189,0
2210 DATA 0,102,0
2220 DATA 0,102,0
2230 DATA 0,102,0
2240 DATA 0,189,0,129,24,129,194,0,67
2250 DATA 228,0,39,232,0,23,240,0,15
2260 DATA 252,0,63,254,0,127,255,0,255
```

Sprites o burbujas
El sprite que definen los 63 bytes de DATA se mueve por toda la pantalla impulsado por las teclas con flecha. Su relativa lentitud es consecuencia del empleo de la rutina en ROM llamada OSWRCH, pero como contrapartida funciona en todos los modos. Al ejecutar el programa (RUN) el listado en assembly desplaza primero la pantalla y después realiza la visualización del gráfico

LLAMA SOFT

Llamasoft es una empresa basada en el talento de Jeff Minter, considerado una estrella en el mundo del software gracias a sus juegos rebosantes de humor e ironía

La mayoría de las casas de software empiezan como "industrias caseras", con uno o dos programadores trabajando en su casa. A medida que van obteniendo éxito, se contratan otros programadores y los diversos factores que la llevaron por el camino del éxito quedan olvidados en la carrera por producir software nuevo; el "estilo de la firma" desaparece y la producción de una empresa se vuelve muy parecida a la de cualquier otra. Pero Llamasoft es diferente: ninguna otra empresa podría haber producido *Revenge of the mutant camels* (La venganza de los camellos mutantes) y *Sheeps in space* (Ovejas en el espacio).

El estilo distintivo de Llamasoft se debe atribuir a un hombre: Jeff Minter. Sólo tiene 22 años de edad pero, en el transcurso de estos últimos años, se ha convertido en una de las estrellas del mundo del software. Es su obsesión por los camellos, las llamas y otros animales peludos la que ha conformado la imagen de Llamasoft y su adjetivo favorito ("¡imponente!") se utiliza habitualmente en la publicidad de Llamasoft. La fama de Minter (o su notoriedad) es tal que ha sido satirizado en uno de los programas de una empresa rival: uno de los peligros del *Manic miner*, de Software Projects, se describe como "El ataque de los teléfonos mutantes".

Minter empezó a escribir juegos en 1981, después de abandonar la Universidad de East Anglia, donde estudiaba matemáticas y física. A pesar de ser el primero de su clase en la parte de su curso relacionada con la informática, no aprobó los exámenes de matemáticas y tuvo que abandonar la carrera.

Su primera incursión en el mercado del software comercial se produjo cuando escribió una versión del *Defender* para el Vic-20 y fundó Llamasoft, en 1982, para comercializar el producto. Pronto vinieron *Traxx* y *Gridrunner*, que le aseguraron el éxito. *Gridrunner*, en especial, se vendió bien, tanto en Gran Bretaña como en Estados Unidos.

En la actualidad Minter escribe juegos para el Vic-20, el Commodore 64 y la gama Atari, pero no para el Spectrum. En consecuencia, de las adaptaciones de los juegos Llamasoft se han hecho cargo Salamander Software y Quicksilva, que producen bajo licencia las versiones para el Spectrum. Llamasoft está por contratar a un ayudante para adaptar juegos, lo que le permitirá a Minter concentrarse en escribir software nuevo, pero seguirá a cargo de la programación. Según explica, "a mí lo único que me interesa es conseguir un buen juego; todos se basan en ideas mías, así que no hay ningún motivo para tomar a otras personas".

La empresa es, en gran parte, un negocio familiar. Llamasoft se convirtió en una sociedad anónima en abril de 1984 y sus directores son el propio Minter, su padre Patrick y su madre Hazel. La empresa se lleva desde la casa familiar de Tadley (Hampshire). Patrick Minter ayuda a su hijo con la programación, mientras Hazel se ocupa de la administración. Llamasoft también tiene empleados a dos ayudantes, dos contables y un director artístico.

Minter lleva producidos 21 juegos para Llamasoft; muchos de éstos contienen un tema recurrente, por lo general con connotaciones de camellos. Minter afirma que él está "totalmente fascinado por los camellos" y el primer juego que incorporó estos animales lo escribió en 1982 para Atari. Se trataba de *Attack of the mutant camels* (El ataque de los camellos mutantes) y fue seguido posteriormente por *Revenge of the mutant camels*.

Llamasoft acaba de firmar un contrato con la CBS para comercializar los juegos de Minter en Gran Bretaña (con la excepción de las adaptaciones de Salamander y Quicksilva). La distribución en Estados Unidos la realiza Human Engineering Software. La demanda de los juegos de Llamasoft es muy grande: un producto nuevo tiene inicialmente una tirada de 10 000 cassettes, y esta tirada se repite luego tantas veces como haga falta, según las ventas registradas.

Minter no tiene previsto ningún cambio respecto a la forma en que funciona la empresa. Afirma que el aspecto comercial no le interesa y planea seguir desarrollando su obsesión por los animales peludos y escribiendo nuevos juegos "imponentes".

La venganza de los camellos mutantes
El surrealista y obsesivo estilo de Llamasoft queda patente en esta fotografía de *La venganza de los camellos mutantes*, continuación del exitoso *Ataque de los camellos mutantes*

Cortesía de "Your 64 and Vic-20"

Editorial Delta, S.A.



10043
9 788485 822836